



62532, 62531, 02312

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

CDIO del 3 - Gruppe 42

Christoffer Perch
Nielsen
s202125



Stig Bødtker Petersen
s186333



Mads Emil Kaas Hansen
s195159



Emil Nymark
Trangeled
s195478



Mohammed Zahed
s186517



Magnus Thorup
Müller Larsen
s194056



Afleveringsfrist 27. november 2020

Link til github repository der blev brugt under projektet:

https://github.com/DTU1semHackerChamps/42_del3

Timeregnskab

Christoffer Perch Nielsen : 35 timer

Emil Nymark Trangeled: 50,5 timer

Stig Bødtker Petersen : 36,5 timer

Mohammed Zahed : 35 timer

Mads Emil Kaas Hansen : 35 timer

Magnus Thorup Müller : 46 timer

Indholdsfortegnelse

Forside	1
Timeregnskab	2
Indholdsfortegnelse	3
Resume	5
Indledning	5
Kravspecifikation	6
Funktionelle krav:	6
Ikke-funktionelle krav:	7
Use cases	8
Use case diagram:	8
Analyse	16
Risikoanalyse	16
Forklaring af begreber:	17
Arv:	17
Abstract:	17
Fortæl hvad det hedder hvis alle fieldklasserne har en landOnField metode der gør noget forskelligt:	17
Klasser:	18
Domænemodel:	19
Systemsekvensdiagram:	20
Design:	21
Designklassediagram:	21
Sekvensdiagram:	22
Implementering	23
Test	24
Dice test	24
Brugertest:	25
Spørgsmål:	25
Hvad synes i om layoutet, altså hvordan spillet ser ud?	25
Hvad synes i om farverne på brættet?	25
Hvad synes i om beskederne?	26
Hvad synes i om spillet som helhed?	26
Andre kommentarer?	26

Test Cases:	27
Screenshot af code coverage:	30
Projektplanlægning	31
Versionsstyring:	31
Google Docs:	31
Git:	32
Konfigurationsstyring	33
Udviklingsmiljøet og produktionsplatformen	33
Github: Til versionsstyring af programmet anvender vi Github.	34
Brugervejledning	35
Hvordan kører man programmet:	37
Importeret af git repo in i IntelliJ:	39
Reload af pom.xml fil:	41
Konklusion	43
Bilag	44
Feltliste:	44
Chancekortliste:	45
Andre kort:	46
Diagrammer og modeller:	46
Litteraturliste	47

Resume

I vores CDIO del 3 projekt har vi lavet et monopoly junior brætspil, hvor man kan være mellem 2 og 4 spillere. I spillet er der 20 forskellige chancekort som har sin egen effekt, der påvirker spillet. Man bevæger sig rundt på en spilleplade med ejendomme, som man kan købe. Spillet går ud på at have så mange ejendomme som muligt, og når en spiller har brugt alle sine penge og går fallit, så er spillet slut. Spilleren med flest penge tilbage på det tidspunkt har vundet spillet.

Indledning

Projektets formål er at skabe et brætspil efter kundens vision. Spillet skal være simpelt at spille, og der skal ikke være behov for særlige instrukser til hvordan man spiller det. Inden vi begynder at skrive selve koden, kigger vi på projektets krav og analysere dem. Dernæst opstiller vi forskellige analyse og design modeller, for at give os et større overblik. Først når det er gjort, begynder vi på koden, hvor vi hen ad vejen går tilbage og ser på vores forskellige modeller og ændrer på dem hvor det er nødvendigt. Derudover laver vi også forskellige tests af koden ved brug af Junit for at sikre os, at vi leverer et ordentligt produkt.

Kravspecifikation

Funktionelle krav:

Spillets krav:

- Der skal være 1 spillebræt.
- Der skal mindst være 2 spillere, men man skal have muligheden for at være op til 4.
- Der skal slås med en terning på en gang
- Spilleplade med felter som har 24 felter.
- Værdien af terningen afgøre, hvor mange felter spilleren rykker frem.
- Spilleren skal indeholde en pengebeholdning.
- Pengebeholdning pr. spiller skal vises.
- Spillerne starter med 20, 18 eller 16 penge afhængig af om der respektivt er 2, 3 eller 4 spiller der er i spillet.
- Vinderen er fundet når den første person går fallit og den spiller med flest penge vinder.
- Hvis der er to spillere, der har lige mange penge, når vinderen skal findes. Skal værdien af spillernes ejendomme lægges til spillerens penge.
- En spiller går fallit når, en spiller ikke har råd til at betale husleje.
- En spiller går fallit når, en spiller ikke har råd til at købe en ejendom.
- En spiller går fallit når, en spiller ikke har råd til at betale afgift fra et chancekort.
- En spillers score må aldrig komme under 0.
- Når en spiller lander på et ledigt felt, skal spilleren købe det.
- Når en spiller har købt en ejendom skal det felt markeres som solgt til den spiller.
- Når en spiller lander på et felt en anden spiller ejer, skal spilleren betale det der står på feltet til feltets ejer.
- Når en spiller lander på et felt, som spilleren selv ejer, skal spilleren ikke betale noget.
- Hvis en spiller ejer begge ejendomme i samme farve er huslejen fordoblet for værdierne der står på felterne.
- Brættet skal indeholde følgende felter: 1 startfelt, 4 chance felter, 1 gratis parkerings felt, 1 gå i fængsel felt, et fængsel felt/på besøg felt og 16 ejendoms felter.
- Når en spiller er nået hele vejen rundt, og lander på start feltet, modtager spilleren 2 penge.
- Når en spiller lander på "Gå i fængsel" bliver spilleren placeret på fængsel feltet.
- En spiller kan komme ud af fængsel ved at betale 1 penge. Derefter skal spilleren kaste terningen for at rykke.
- En spiller skal komme ud af fængsel ved at bruge "du løslades uden omkostninger" - kortet, hvis spilleren har det. Derefter skal spilleren kaste terningen for at rykke.
- En spiller kan modtage husleje mens spilleren er i fængsel.
- Spillet skal indeholde 20 chancekort.
- Der udskrives en tekst når man lander på et felt.

- Spillerne skal beholde deres position efter de har landet på et felt.

Vores krav:

- Der skal være en mulighed for at genstarte spillet efter en spiller har vundet spillet.

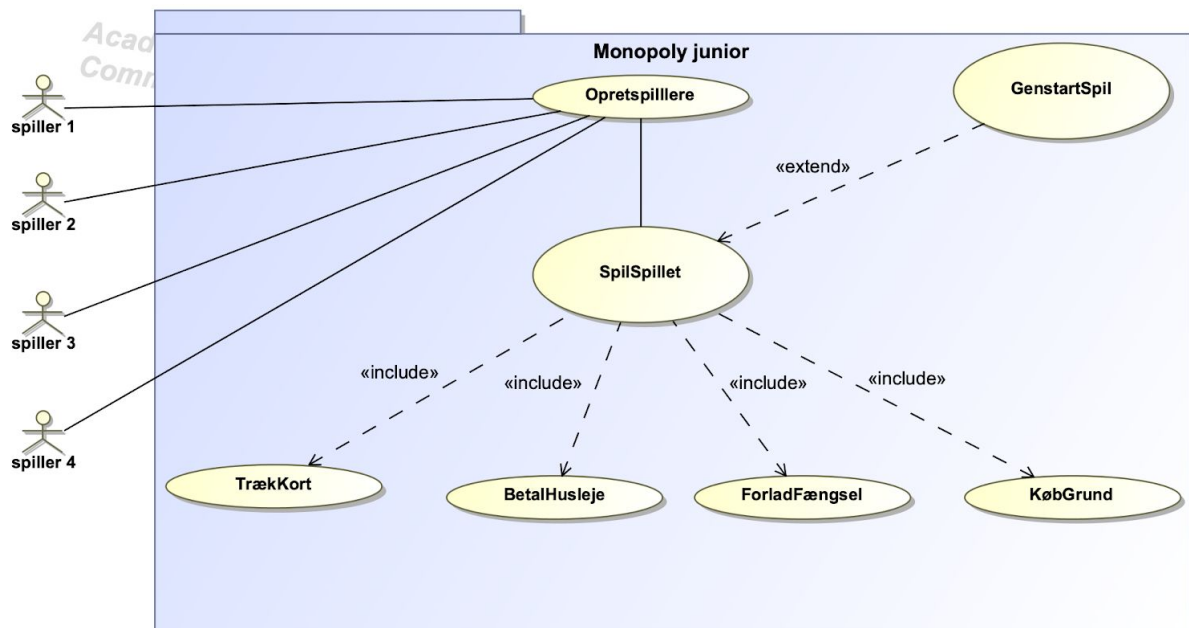
Ikke-funktionelle krav:

- Spillet skal kunne spilles på DTU's databaser.
- Kunden skal bruge JDK 1.8 +
- Kunden skal bruge Windows 10 +
- Kunden skal bruge en java compiler.
- Programmet testes i Junit
- Programmet skrives i IDE'en IntelliJ version 2020.2
- Tegnsættet i projektet er UTF-8

Use cases¹

- SpilSpillet
- GenstartSpil
- TrækKort
- KøbGrund
- OpretSpillere
- BetalHusleje
- ForladFængsel

Use case diagram:²



¹ https://docs.google.com/document/d/1-XGd8bKliHMJeK0ILX8EHQ7FTbhvrboE7ldwqu_zQbQ/edit

² https://drive.google.com/file/d/1z72iQwUe4cxd-HaHHm98m0lpLckVI2_z/view?usp=sharing

Use case: SpilSpillet
ID: 1
<p>Brief description:</p> <p>Efter man starter spillet, bliver man bedt om at vælge hvor mange spillere man er (2-4) og derefter bliver spillerne oprettet og så kan man gå i gang med spillet.</p>
<p>Primary actors:</p> <p>Spiller 2-4.</p>
<p>Secondary actors:</p> <p>Ingen.</p>
<p>Preconditions:</p> <p>Der skal være minimum 2 spillere for at spille spillet .</p>
<p>Main flow:</p> <ol style="list-style-type: none"> 1. Man trykker spil på startside. 2. Derefter genererer spillet 2-4 spillere, så man kan skiftes om at slå med terningerne. 3. Spillerne lander på et felt på brættet baseret på hvad de slog med terningerne. 4. Spillerne lander på et felt og effekt sker baseret på hvilket felt de lander på <ol style="list-style-type: none"> a. Spiller skal købe en ledig grund. b. Spiller skal betale husleje c. Spiller skal trække et chance kort d. Spiller ryger i fængsel e. Spilleren lander på sin egen grund og skal ikke gøre noget. f. Hvis spilleren lander på eller passerer startfeltet modtager man 2 penge 5. Spillet fortsætter indtil der er en spiller som går fallit. Derefter har den spiller med flest penge vundet spillet.
<p>Postconditions:</p> <p>Spillet erklærer en vinder, når den første person går fallit.</p>
<p>Alternative flows:</p> <p>Hvis to spiller har samme mængde penge, så skal værdien af deres grunde lægges til deres pengebeholdning, for at se hvem der har vundet.</p>

Use case: GenstartSpil
ID: 2
Brief description: Spillerne genstarter spillet, når de har lyst til.
Primary actors: Spiller 1
Secondary actors: Ingen.
Preconditions: <ol style="list-style-type: none"> 1. Spillerne skal have startet spillet, før de kan genstarte spillet. 2. Der skal være erklæret en vinder, som sker når en person går fallit, så ser man hvem har flest penge.
Main flow: <ol style="list-style-type: none"> 1. Use casen starter når spillerne er klikket til genstartspil. 2. Spillet starter helt forfra, når de klikker gestartspil.
Postconditions: Systemet sletter gamle score og starter et helt nyt spil.
Alternative flows: Ingen

Use case: TrækKort
ID: 3
Brief description: Hvis man lander på et chancekort, så trækker spilleren et chance kort.
Primary actors: Spiller 2 - 4
Secondary actors: Ingen.
Preconditions: <ul style="list-style-type: none"> • Spillet er Startet • En spiller har landet på et chancefelt
Main flow: <ol style="list-style-type: none"> 1. Spilleren lander på chance felt 2. Spilleren trækker et chancekort 3. Spilleren udfører handlingen på kortet
Postconditions: Spilleren eller spillerne har gjort det der er specificeret på chancekortet
Alternative flows: Ingen.

Use case: KøbGrund
ID: 4
Brief description: En spiller lander på en ledig grund, og skal købe den.
Primary actors: Spiller 2 - 4.
Secondary actors: Ingen.
Preconditions: <ul style="list-style-type: none"> • Spillet er Startet • Spilleren har landet på en ledig grund
Main flow: <ol style="list-style-type: none"> 1. Spilleren betaler den mængde af penge, som grunden kræver
Postconditions: Spilleren ejer den købte grund.
Alternative flows: <ol style="list-style-type: none"> 1. Hvis spilleren har ikke råd til grunden, er spilleren gået falit 2. Hvis der trækkes et chancekort, som gør at man kan rykke hen til enhver plads, og der ikke er nogen ledige grunde. Kan man købe en grund fra en anden spiller.

Use case: OpretSpiller
ID: 5
<p>Brief description:</p> <p>Inden selve spillet starter, vælger man antal spillere, hvor 2 er minimum og 4 er maksimum.</p>
<p>Primary actors:</p> <p>Spiller 2-4.</p>
<p>Secondary actors:</p> <p>Ingen.</p>
<p>Preconditions:</p> <ul style="list-style-type: none"> • Man har åbnet spillet
<p>Main flow:</p> <ol style="list-style-type: none"> 1. Man klikker på det antal spillere, man har brug for. 2. Vælger hvilken karakter man vil være 3. Spillet går i gang
<p>Postconditions:</p> <p>Spillerne er oprettet, og man er klar til at spille.</p>
<p>Alternative flows:</p> <p>Ingen.</p>

Use case: BetalHusleje
ID: 6
Brief description: En spiller har landet på en grund som allerede er ejet og skal betale husleje.
Primary actors: Spiller 2 - 4.
Secondary actors: Ingen.
Preconditions: En spiller har landet på en ejet ejendom
Main flow: <ul style="list-style-type: none"> • Spiller betaler husleje til grundens ejer.
Postconditions: <ul style="list-style-type: none"> • spilleren kan fortsætte med at spille.
Alternative flows: <ul style="list-style-type: none"> • Spilleren ejer selv den ejendom og skal ikke betale noget. • Spilleren kan ikke betale og går fallit.

Use case: ForladFængsel
ID: 7
Brief description: Betale sig ud af fængslet, eller brug forlad fængsel chancekort
Primary actors: Spiller 2 - 4.
Secondary actors: Ingen.
Preconditions: <ul style="list-style-type: none"> • En spiller sidder i fængsel
Main flow: <ol style="list-style-type: none"> 1. Spilleren betaler sig ud af fængslet 2. Spilleren kaster sin terning
Postconditions: Spilleren kan fortsætte med spillet, efter løsladelsen.
Alternative flows: Hvis man har chance kortet, der får en ud af fængslet, kan man bruge kortet for at komme ud i stedet for at betale.

Analyse

Vores diagrammer og modeller findes også under bilag, hvor man bliver ført hen til et sted hvor man bedre kan se det.

Risikoanalyse³

- Undervurderet projekt størrelse
- Planlægning af tid
- Stor design omstrukturering
- Fejlet system integration
- Ikke leveret system dele
- Misforstået krav
- Sygdom blandt holdmedlemmer
- Manglende erfaring

Risici	Sandsynlighed	Skade	Risiko
Undervurderet projekt størrelse	1	5	5
Planlægning af tid	4	5	20
Stor design omstrukturering	1	2	2
Fejlet system integration	4	2	8
Ikke leveret system dele	2	5	10
Misforstået krav	1	1	1
Sygdom blandt holdmedlemmer	5	2	10
Manglende erfaring	4	5	20

Planlægning af tid:

Forebyggelse: Vi holder møder i gruppen, aftaler hvad der skal laves og til hvilket tidspunkt.

Aktion: Vi er nødt til at lave stramme deadline, som skal overholdes, ellers får det konsekvenser for den enkelte person.

Manglende erfaring:

Forebyggelse: Vi læser op på tavle-noterne, så godt vi kan.

Aktion: Vi kontakter en hjælpelærer.

³ CDIO 1 -

https://docs.google.com/document/d/1-XGd8bKliHMJeK0ILX8EHQ7FTbhvrboE7ldwqu_zQbQ/edit

Forklaring af begreber:

Arv:

Når man bruger arv i sin kode, betyder det at en klasse nedarver attributter og metoder fra en anden klasse. Der er altså en relation mellem klasserne. I vores tilfælde, bruger vi arv til at lave vores chancekort i vores kode. Derfor har vi lavet en klasse, som hedder ChanceCard, hvor vi definerer cardDescription, tiles, players, currentPlayer og gui. Det kan vi så bruge, når vi skal arbejde med de specifikke chancekort, som f.eks. i specificMoveCard hvor man kan vælge at rykke op til 5 gange. Her nedarver vi ChanceCard klassen ved at bruge “extends”

```
public class specificMoveCard extends ChanceCard {
```

Derefter har vi brugt “super” for at kalde på vores ChanceCard konstruktør.

```
public specificMoveCard(String cardDescription, Tile[] tiles, Player[] players, Player currentPlayer, GUI gui) {  
    super(cardDescription, tiles, players, currentPlayer, gui);  
}
```

Abstract:

En abstrakt klasse er en klasse hvor der ikke kan oprettes objekter i, og det skyldes at klassen ikke er instantieret. I den abstrakte klasse er der metoder, som ikke er implementeret og hvis man ønsker at lave en abstrakt klasse kan det skrives ligesom her:

```
public abstract class ChanceCard {  
  
    protected String cardDescription;  
    protected Tile[] tiles;  
    protected Player[] players;  
    protected Player currentPlayer;  
    protected GUI gui;  
}
```

Abstrakte klasser bruger man når man arbejder med nedarvning, da man kan nedarve klassen som har en “body” hvorefter man så implementerer metoderne, som man vil bruge.

Fortæl hvad det hedder hvis alle fieldklasserne har en landOnField metode der gør noget forskelligt:

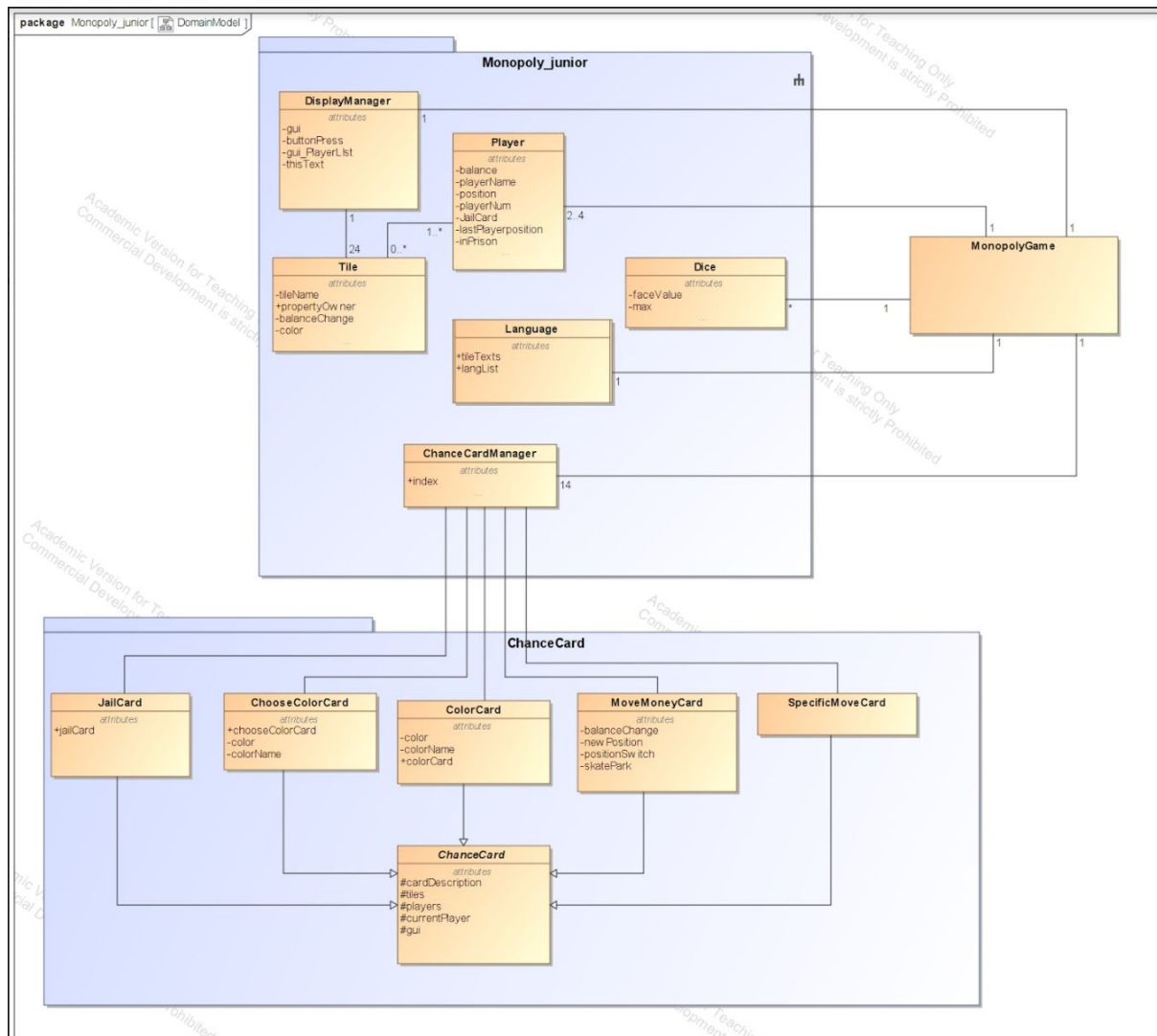
Det kaldes polymorfi, som betyder “mange former”. Det er et begreb som anvendes inden for nedarvning, da det gør det muligt at bruge de metoder som vi har nedarvet.

Klasser:

Vi har defineret følgende klasser i disse forskellige packages:

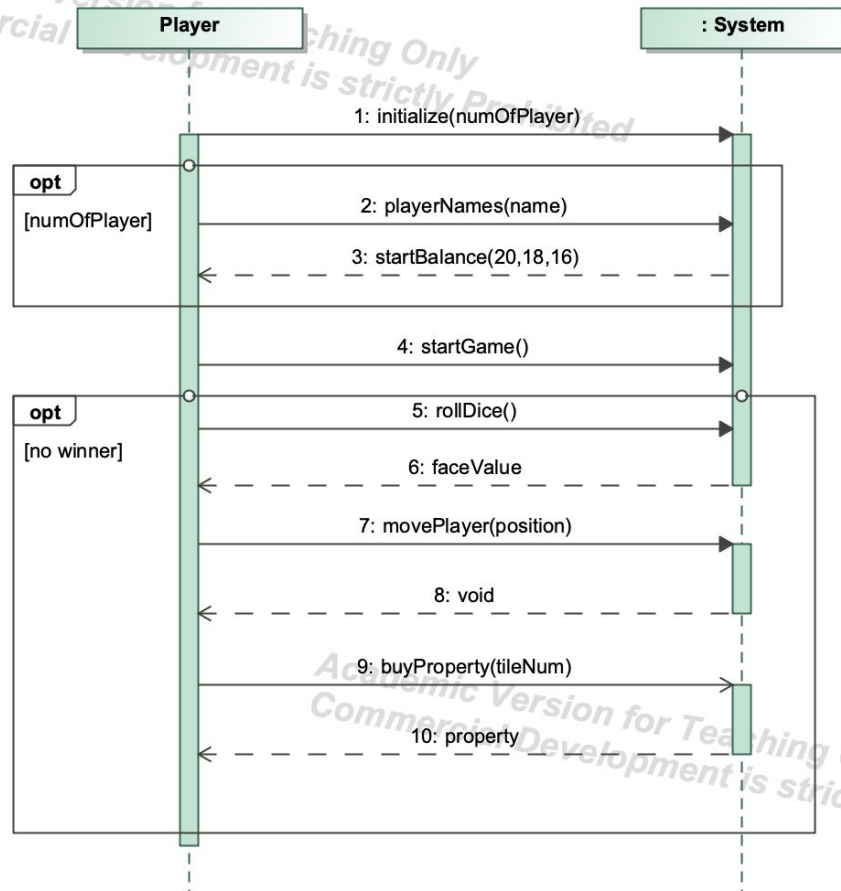
- Monopoly_junior Package
 - ChanceCardManager
 - Dice
 - Displaymanager
 - Language
 - Player
 - Tile
- ChanceCard Package
 - ChanceCard
 - CharacterCard
 - ChooseColorCard
 - ColorCard
 - JailCard
 - moveMoneyCard
 - specificMoveCard
- Test Package
 - DiceTest
 - LanguageTest
 - PlayerTest
 - TileTest

Domænemodel:⁴



⁴ <https://drive.google.com/file/d/1Ua4FJA6-ud4EZ5r9jf-fFRy2nMk3uulA/view?usp=sharing>

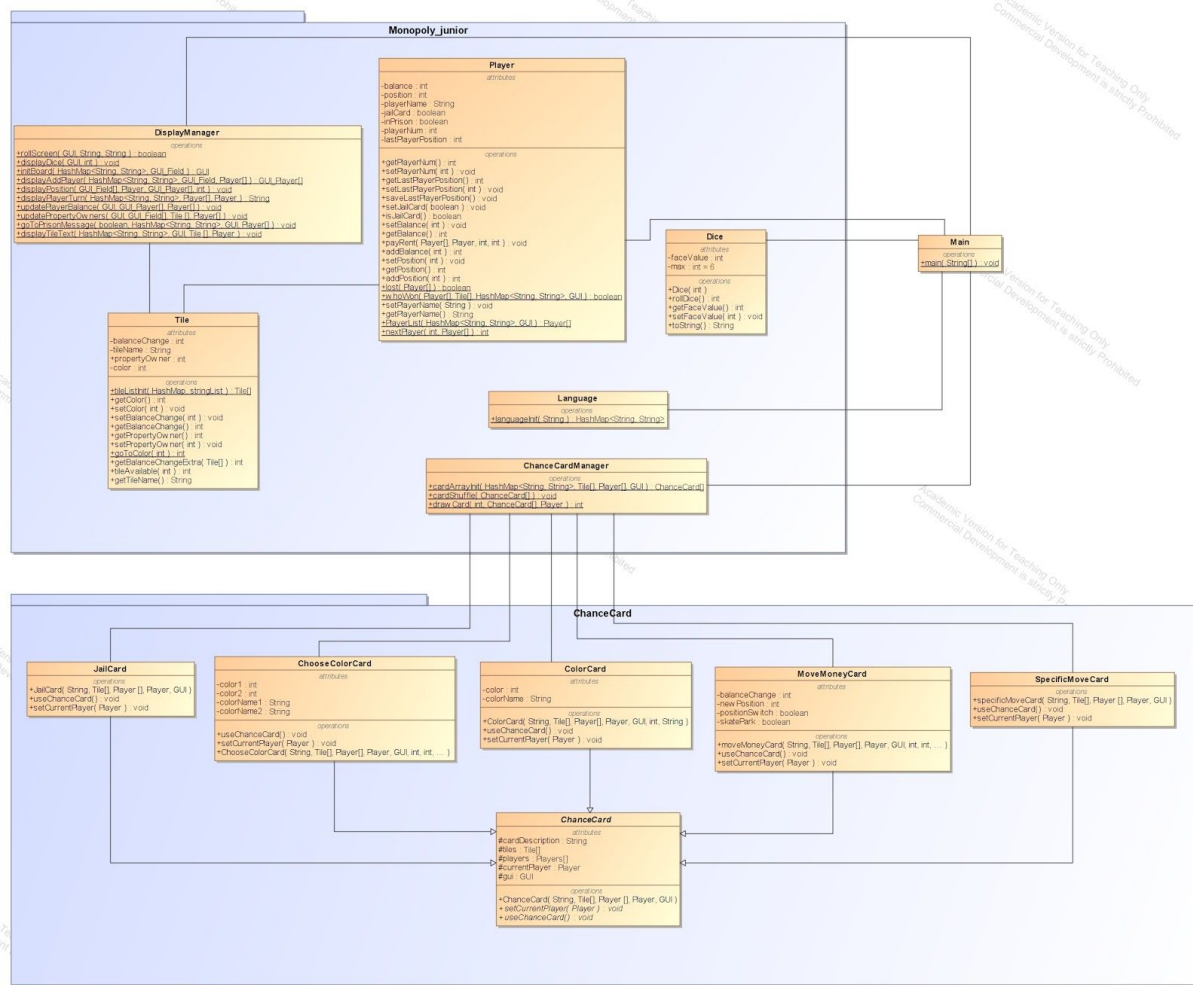
Systemsekvensdiagram:



Vores diagrammer og modeller findes også under bilag, hvor man bliver ført hen til et sted hvor man bedre kan se det.

Designklassediagramm:⁵

<https://drive.google.com/file/d/1J8qnNspF321LNA9tJqCpzev6uCtp0j0J/view?usp=sharing>



Det ses på diagrammet at vi har brugt **GRASP**. Vi har høj samhørighed og lav coupling. Den høje samhørighed ses på de enkelte klasser, med deres attributter og metoder, som beskriver hvad de gør hver især. Lav coupling ser vi ved at der er få koblinger mellem de enkelte klasser som f.eks. kan det ses på de klasser der ligger i monopoly junior pakken. Dog er der en noget højere coupling på vores “ChanceCardManager”, det skyldes nedarvning og at “ChanceCardManageren” er en information expert for alle de klasser der er nedarvet fra den abstrakte klasse “ChanceCard”. Det kan også ses, at vi har en “Displaymanager”, som holder styr på de mest centrale metoder der vedrører den grafiske brugergrænseflade, som altså også gør den til en information expert.

⁵ <https://drive.google.com/file/d/1J8gnNspF321LNA9tJqCpzev6uCtp0j0J/view?usp=sharing>

Sekvensdiagram:

Vores sekvensdiagram er så stort, at vi har valgt kun at dele det som et bilag/link:

<https://drive.google.com/file/d/1BVX5oqcqU60zP0tIJx2dhDx17eEge-a/view?usp=sharing>

Implementering⁶

I dette projekt har vi været et behov for at prioritere hvilke elementer det skal implementeres i det endelige produkt. Vi har gået med den tilgang. Vi tog den tilgang at vi analyseret hvilke elementer som vi følte hvilke gav mest til spillets helhed, derefter undersøgte vi hvor mange ressourcer de elementer krævede for udførsel, dette ledte til at vi nedprioriterede ting så som, de kortene vi kalder for characterCards, som var at man skal give kortet til en bestemt spiller, hvor så den spiller skal udføre det der står på kortet. dette gjorde vi fordi vi følte de ville tage mange ressourcer at udføre, hvor vi mente at hvis vi fik de resterende 14 kort i en virkende tilstand, ville det give et mere tilfredsstillende spil. Vi valgte også at nedprioritere reglen med at den yngste starter spillet, da vi følte at i helheden af spille ville det ikke tilføje mest til spillet, og vi følte også at den regel ville kunne løses af spillerne før spillet starter.

Vi har også valgt at nedprioritere 2 andre kort, grundet de begrænsede ressourcer.

I projektet har vi brugt Git, og ligeledes Github. Vi har valgt at have en branch struktur med main, developer og feature-branches, hvor alle features bliver udgrenet fra developer branchen. Vi har levet feature-branches ud fra klasser vi fandt der skulle bruges til spillet.

Når en feature blev færdig, vælger vi at merge den pågældende feature ind i developer-branchen, eller andre feature branches hvor der er behov. Når developer-branchen, så er forholdsvis bug fri og fungerer som forventet, bliver det til sidst merge ind i main-branchen. Vi har også benyttet os af muligheden for at bruge GUI'en, som er lavet til projektet.⁷

⁶

<https://docs.google.com/document/d/1npugHuQcEa7tSAv3d7KhBL3mh1j5qmquirQG5cY3m8/edit#heading=h.65fze1prdg3f>

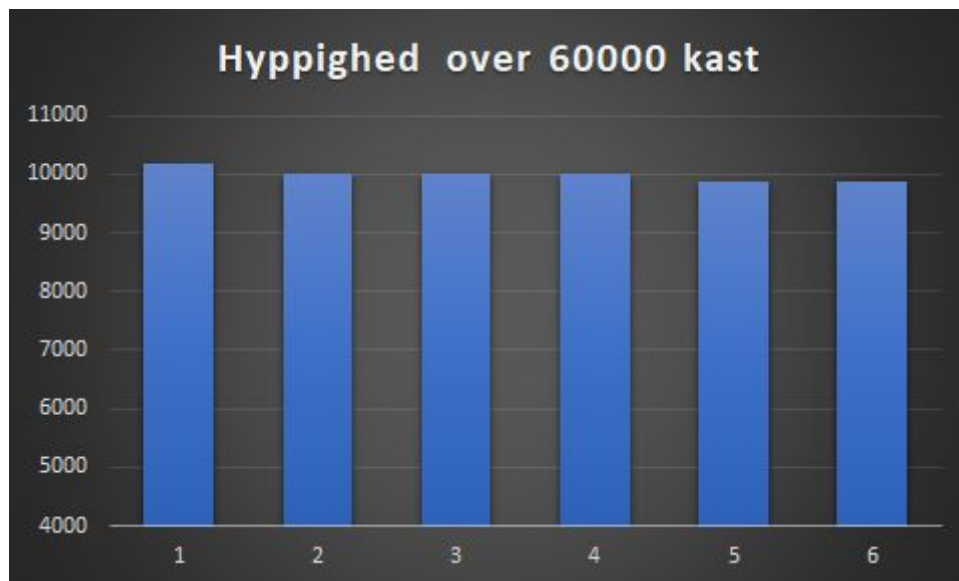
⁷ CDIO 1 -

https://docs.google.com/document/d/1-XGd8bKliHMJeK0ILX8EHQ7FTbhvrboE7ldwqu_zQbQ/edit?usp=sharing

Test

Dice test⁸

I denne test er der blevet slået med en 6 sided terning 60000 gange via vores program. Resultaterne ses nedenunder og giver os en indikation af at det fungerer, da der er lige så stor chance for at slå 1 som er der for at slå 6. Der vil altid være held involveret når man kaster med en terning og derfor har slået terningen 60000 gange. I vores test med Junit har vi også givet testen et delta på 400, som “passed”.



Resten af testene til projektet er blevet udført ved hjælp af Junit 5, hvor vi har testet, at programmet trækker og giver de rigtige antal point afhængig af hvor man lander på spillebrættet. Derudover har vi også testet at vores extraturn funktion virker. Da terningen er en stor del af spillet, fandt vi det også relevant at teste, at terningen kun kunne slå mellem 1-6, og at sandsynligheden for at slå 1-6 var ca. den samme.

Vi har også lavet en test der viser, at ens balance aldrig kan komme under 0.

Derudover har vi lavet nogle tests andre tests:

- spillerens position
- vores “win” metode virker
- test af vores shiftPlayer
- test af language klassen

8

<https://docs.google.com/document/d/1npugHuQcEa7tSAv3d7KhBL3mh1j5qmquirQG2cY3m8/edit#heading=h.65fze1prdg3f>

Brugertest:

Testpersoner: Emils mor og far

De har ikke fået reglerne at vide til at starte med.

Starten gik nemt. De fik hurtig valgt spillere og rullede med terningen med det samme. De blev lidt forvirrede over at karaktererne var biler og ikke det som navne beskriver.

De er forvirrede over hvad der sker på spillebrættet, når de lander på et chancekort. De prøver trykke på det store spørgsmålstegn i midten for at trække et chance kort. De prøver at trykke på "OK" knappen i stedet. Efter ca. 1 minut fandt de ud af at de skulle trykke på OK knappen. De tror at man skal klikke på felterne for at rykke frem til dem, istedet for at trykke på OK for at rykke.

Efter de begge havde prøvet at rulle med terningen 2 gange. Fik de lært at man bare skulle klikke på OK for at gå videre. De lader musen gå på omgang, så når det er deres tur giver de musen videre.

Sådan fortsatte det.

Emils mor synes at det er et dårligt spil, fordi hun tabte.

De er lidt usikre på hvem der er spiller 1, når spillet spørger om de vil genstarte.

De ved ikke, at man betaler for at komme ud af fængslet

De er lidt i tvivl om hvordan og hvornår pengene kommer ind i deres pengebeholdning og ud igen.

Nu får de læst reglerne til spillet.

Spørgsmål:

Hvad synes i om layoutet, altså hvordan spillet ser ud?

Mor: Figurerne er lidt skuffende i forhold til dem, som der er på spil manualen
Vinduet må gerne være større og det er svært at læse

Far: Layoutet er overskueligt.

Figurerne må gerne være afspejlet i forhold til navnet af karakteren

Hvad synes i om farverne på brættet?

Mor: Den lyseblå og brune farve er forstyrrende. Ville blande farverne, så de ikke står ved siden af hinanden. Sådan er det nemlig i matador.

Far: Retro agtigt.

Hvad synes i om beskederne?

Mor:

Det er nogle gode tekster, det er nemt at forstå, hvad man skal.

Far: Er enig med mor.

Hvad synes i om spillet som helhed?

Mor: Man skal nok være flere end 2 for at få den fulde oplevelse. Så man når at vide hvad spillet går ud på. Nåede aldrig at finde ud af hvad det store spørgsmålstegn i midten gik ud på.

Far:

Man læste beskederne istedet for at se, hvor man røg hen på brættet.

Andre kommentarer?

Ingen andre kommentarer fra testpersonerne.

Test Cases:

Project Name: CDIO_del3

Test Case ID	TC01	Test Designed by	Group 42
Test Case Description	Test for player balance	Test Executed by	Md. Zahed
Test Priority	High	Test Execution date	26/11-2020
Test Execution Description:	Execution of the test has been done through Junit-test (Junit 4).	Environment	macOS

Precondition:

-Non

Purpose of the test:

- To make sure that, balance of a player never comes below 0.

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Balance is provided to player	5	1	18	Pass	Player had 13 in beforehand
2	Balance is provided to player	-11	0	0	Pass	Player had 10 in beforehand
3	Balance is provided to player	-10	0	0	Pass	Player balance was set to -15

Post-conditions: Non

Project Name: CDIO_del3

Test Case ID	TC02	Test Designed by	Group 42
		Test Executed by	Md. Zahed
Test Priority	Medium	Test Execution date	26/11-2020
Test Execution	The execution happened through Monopoly-game.	Environment	macOS

Test Case Description:

- Creation of players.

Precondition:

- Player creates a player (in the game) with a name.

Purpose of the test:

- Visibility of the created-player name along with start-balance (which is 20, 18 or 16 depends on number of the players).



Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	- a game is created with two players. - Entered [name]	Player1 Player2	Player1 Player2	Player1 Player2	Pass	Player names and start balance are visible
2	- another game is created with two player option. - Entered [name]	Observer1 Observer2	Observer1 Observer2	Observer1 Observer2	Pass	Player names and start balance are visible

**Post-conditions:**

- Players|should be able to see their name beside their balance.

Project Name: CDIO_del3

Test Case ID	TC03	Test Designed by	Group 42
Test Case Description	A test of all scenarios of the game.	Test Executed by	Md. Zahed
Test Priority	High	Test Execution date	27/11-2020
Test Execution Description:	Monopoly junior game is played by a person as role of 2 players.	Environment	macOS

Precondition:

- Minimum two or maximum four players are required in order to create a game.

Purpose of the test:

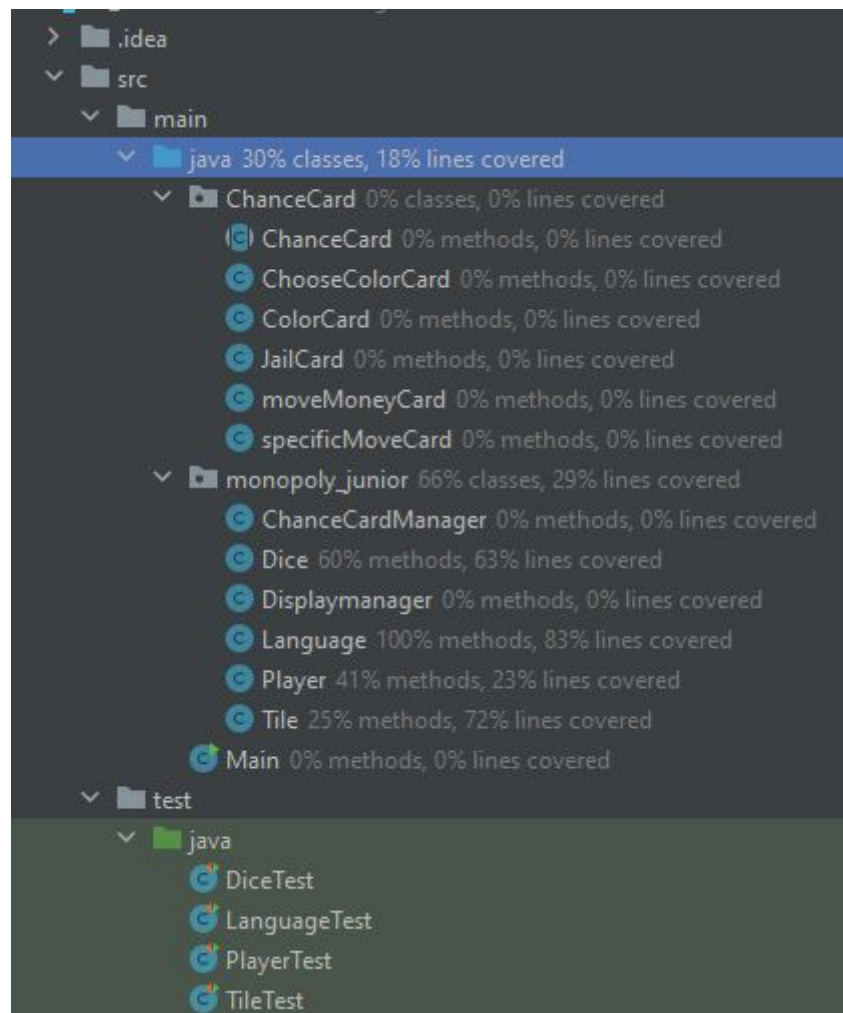
- Character names and start balance.
- Whether or not, the property ownership is determined by their landing position.
- Free landing on an own land.
- Properties
- Winner of the game.

1	Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
2	A two player-game is created.	Character name (Car and Cat)	Character name (Car, Cat) Start balance 20	Character name (Car, Cat) Start balance 20	Pass	Both players have 20 in hand
3	Cat rolls and gets 3 as a face-value	Face-value 3	Property (Chance card and a random property). Balance 20	Chance card. An option to choose Candy and Ice cream store. Balance 20	Pass	Cat lands on Chance-card with a face-value of 3. Now Cat is owner of Ice cream shop. Chance-card costs 0
4	Car rolls and gets face-value 4.	Face-value 4	Candy store & balance 19	Candy store & balance 19	Pass	Candy store costs 1, and balance of Car is now 19.
5	Cat rolls and lands on free parking.	Free-Parking land	Balance 18	Balance 18	Pass	Cat has had 18 beforehand. Free-parking land costs 0
6	Cat gets 1 as a face-	Property	Zoo & balance 9	Zoo & balance 9	Pass	Cat's position is Bolling

	value	ownership				alley and has had 9 in hand.
7	Car lands on Boardwalk with a balance of 4	Balance (-5)	Car's balance 0 and Cat as a winner.	Car's balance 0. Player 2 wins	Fail	Cat has 12 in hand.

Screenshot af code coverage:

Vi har brugt “code coverage” i IntelliJ for at dokumentere hvor meget at vores kode som vi har testet:



Projektplanlægning

Versionsstyring:

I dette projekt har vi brugt git, og github til versionsstyring af vores kode, og google docs til rapportskrivning og ligeledes holde versionsstyring af den.

Vi startede projektet med at analysere kundes krav og derfra udarbejde og definere vores kravspecifikation. da projektet ikke har ændret sig meget har vi genbrugt vores use cases og opdateret dem så de passer med vores nye krav. Vi brugte det til at udarbejde vores use case diagram.

Vi genbrugte vores risici analyse fra vores CDIO del 1 opgave, da vi følte at der ikke var store ændringer i et aspekt. Selvfølgelig tog vi de samme forbehold og forebyggende på de områder vi mente havde højest risici for projektets fuldførelse.

Derefter påbegynde arbejdet på vores domænemodel, hvilket vi brugte til at opbygge vores klassesdiagram.

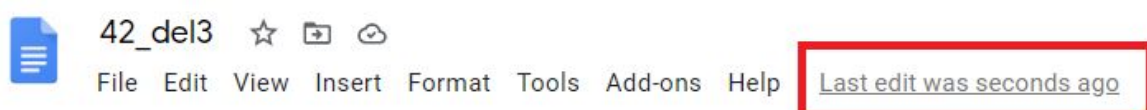
Da vi nu havde fundet de klasser og metoder vi mente vi skulle bruge. følte vi at vi kunne gå i gang med implementeringen af koden.

Vi uddelegerede arbejdsopgaverne rigeligt mellem gruppemedlemmerne og påbegynde implementeringen af vores kode.


Efter klasserne var lavet begynde vi med vores test af spillet.

Google Docs:

Versionsstyring af Google Docs, her har vi ikke sat nogle specielle regler for håndtering af versionsstyringen da den allerede tilbyder en flexibel arbejdsmetode. Man ville kunne finde Versions Historikken ved at klikke på knappen “Version history”:




Dette ville tage en ind til historikken, hvor man kan se ændringerne ude i højre side, og revert tilbage til en tidligere version, hvis der skulle være brug for det.




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

CDIO del 3 - Gruppe 42


Christoffer Perch Nielsen
s202125




Emil Nymark Trangeled
s195478



Stig Bodtker Petersen
s186333



Mohammed Zahed
s186517



Total: 2 edits

Version history

Only show named versions

TODAY

26 November, 11:23
Current version
Christoffer Nielsen
Stig Petersen

TUESDAY

24 November, 19:16
Christoffer Nielsen

LAST WEEK

18 November, 13:00
Christoffer Nielsen

16 November, 14:11
Zee28

THIS MONTH

14 November, 21:29
Stig Petersen

13 November, 23:01
Djuncle

13 November, 10:10
Zee28
Dunno Afrid

12 November, 14:45
Djuncle
Trangeled

Show changes

Git:

Versionsstyring af kode, her har vi brugt git, og ligeledes github. her har vi sat regler op for generel brug.

Den første regel er at der må max være en der arbejder på en klasse af gangen. Dette er gjort for at holde merge conflicter til et minimum, og er forholdsvis nemt at overholde i dette lille projekt, med relativt få udviklere.

Den næste regler er for brug af git.

Vi har valgt at have en main branch hvorpå alt kode skal virke i en form for release candidate. her må der ikke skrives nogen form for kode.

Ud Fra main branchen har vi developer branchen, her må der godt være kode som ikke virker, dog må der stadig ikke skrives nogen form for kode i denne branch, denne branch skal blive brugt til til merging til og fra for andre branches som bliver udgrenes fra denne branch.

Herfra skal alt kode der skal tilføjes ske via at lave en ny brach ud fra developer branchen. Hver branch der vil blive lavet vil være en feature branch. En feature branch skal have en bestemt navngivning, navngivningen skal foregå følgende: det første der skal stå er hvad featuren er efterfulgt med et underscore og så feature, så Fx. Dice_feature. Når den branch er lavet må der kun laves kode der er relevant inden for den feature, alt andet skal have sin egen feature branch.

Konfigurationsstyring

Udviklingsmiljøet og produktionsplatformen

Udviklingsmiljø: Udviklingsmiljøet vi bruger til at udvikle projektet, er IntelliJ IDEA Ultimate Version 2020.2.4. Den kan downloades [her](#).

Download IntelliJ IDEA

[Windows](#) [Mac](#) [Linux](#)

Ultimate
For web and enterprise development

Download .exe ▼

Free 30-day trial

Community
For JVM and Android development

Download .exe ▼

Free, open-source

JDK version: Vi bruger Java JDK version 1.8.0_271 som vores projekt SDK og Language level 8 som vores projekt language level. Java SE Development Kit 8 kan downloades [her](#).

Windows x86	154.48 MB	jdk-8u271-windows-i586.exe
Windows x64	166.79 MB	jdk-8u271-windows-x64.exe

Inde under File > Project Structure i IntelliJ kan man så konfigurere Project SDK'en og Project Language Level hvilket vi har gjort her.

Project name:
42_del3

Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.
1.8 java version "1.8.0_271" Edit

Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.
8 - Lambdas, type annotations etc.

Github: Til versionsstyring af programmet anvender vi Github⁹.

DTU1semHackerChamps / 42_del3 Watch

<> Code ⓘ Issues 🔗 Pull requests ▶ Actions 📁 Projects 📖 Wiki 🛡 Security 📈 Insights

🔗 main 8 branches 0 tags Go to file Add file Code

emilnt1 Test names removed ae9b9a1 14 days ago 12 commits	
.idea	Test names removed 14 days ago
src/main/java	Test names removed 14 days ago
.gitignore	Created .gitignore 14 days ago
42_del3.iml	wrote name.. 14 days ago
pom.xml	Added project with maven and imported GUI with test in main 14 days ago

Help people interested in this repository understand your project by adding a README. Add a README

Maven: For at gøre byggeprocessen af vores program nemmere anvender vi Maven, der blandt andet gemmer vores GUI artifacts. Til GUI'en vi har fået udleveret i undervisningen matadorgui.jar bruger vi altså dertil Maven til at gøre det nemmere at køre den.

JUnit: Til tests af vores program, anvender vi JUnit 5.4.2. Man kan importere JUnit 5.4.2 direkte inde i IntelliJ ved at gå ind i Test, trykke F2 og Importere til IntelliJ.

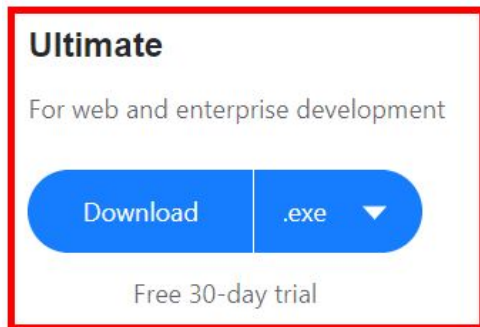
Styresystem version: Programmet bliver skrevet på Windows 10 version 2004 build 19041.630.

Rapportskrivning: Til rapportskrivningen anvender gruppen Google Docs, da det er nemt, hurtigt og effektivt til rapportskrivning.

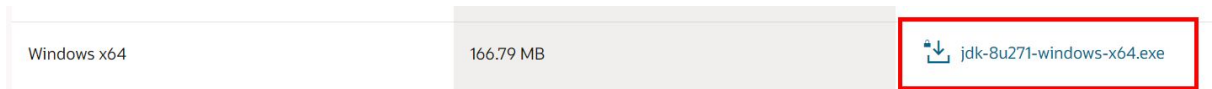
⁹ [DTU1semHackerChamps/42_del3 \(github.com\)](https://github.com/DTU1semHackerChamps/42_del3)

Brugervejledning¹⁰

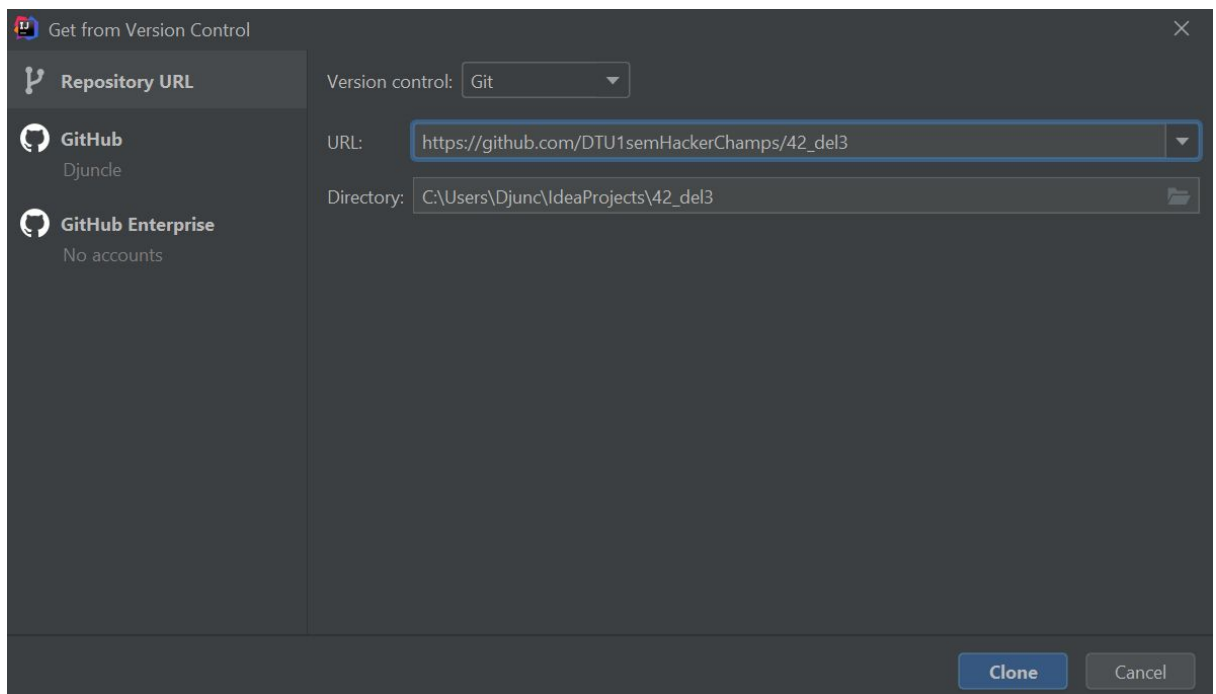
1. Brugeren skal først downloade og installere IntelliJ IDE'en. Den kan downloades [her](#).



2. Derefter skal brugeren downloade og installere en JDK. Vi bruger JDK 1.8. Den kan downloades [her](#). Man skal være logget ind på Oracle.



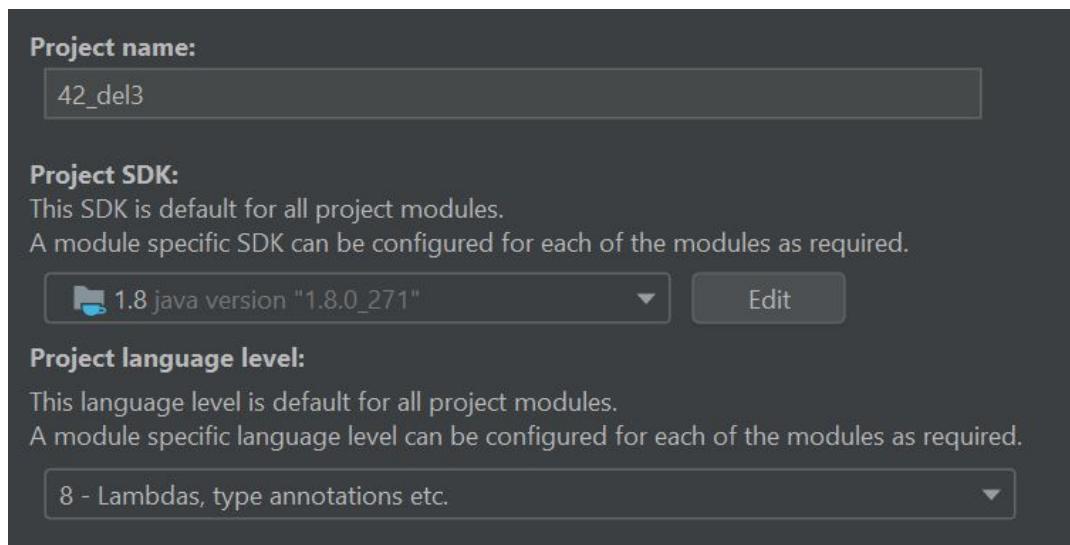
3. Brugeren downloader .zip filen fra Github repositoriet eller importere det direkte ind i IntelliJ ved at bruge File -> New -> Project from Version Control. Inde på URL boksen skal man så paste linket til [Github repositoriet](#) og derefter skal man trykke på "Clone".



¹⁰ Vi har genbrugt dele af vores brugervejledning fra CDIO 2
<https://docs.google.com/document/d/1npugHuQcEa7tSAv3d7KhBL3mh1j5qmquirQG2cY3m8/edit?usp=sharing>

Gem den i et directory som du selv vælger. Her er den gemt i IdeaProjects mappen. Hvis man aldrig har brugt Github før kan man gå længere ned og se en mere detaljeret tutorial.

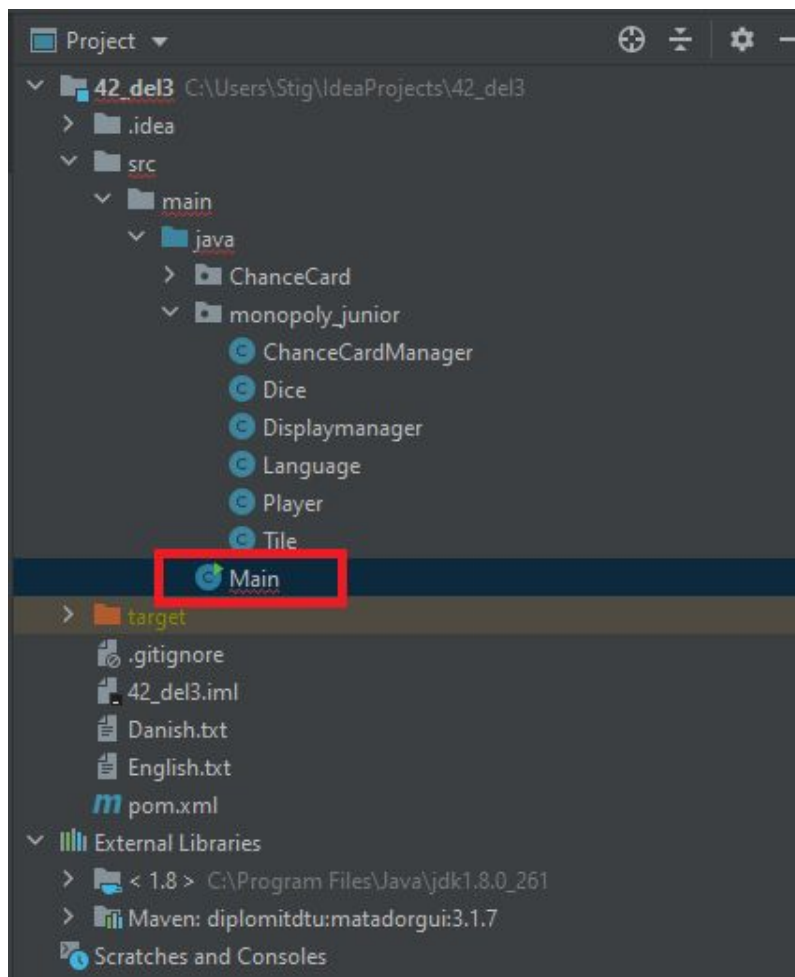
4. For at sikre sig at projektet virker skal man nu under File > Project Structure gå ind og ændre Project SDK'en til JDK 1.8 version 1.8.0_271 og Project language level til 8.

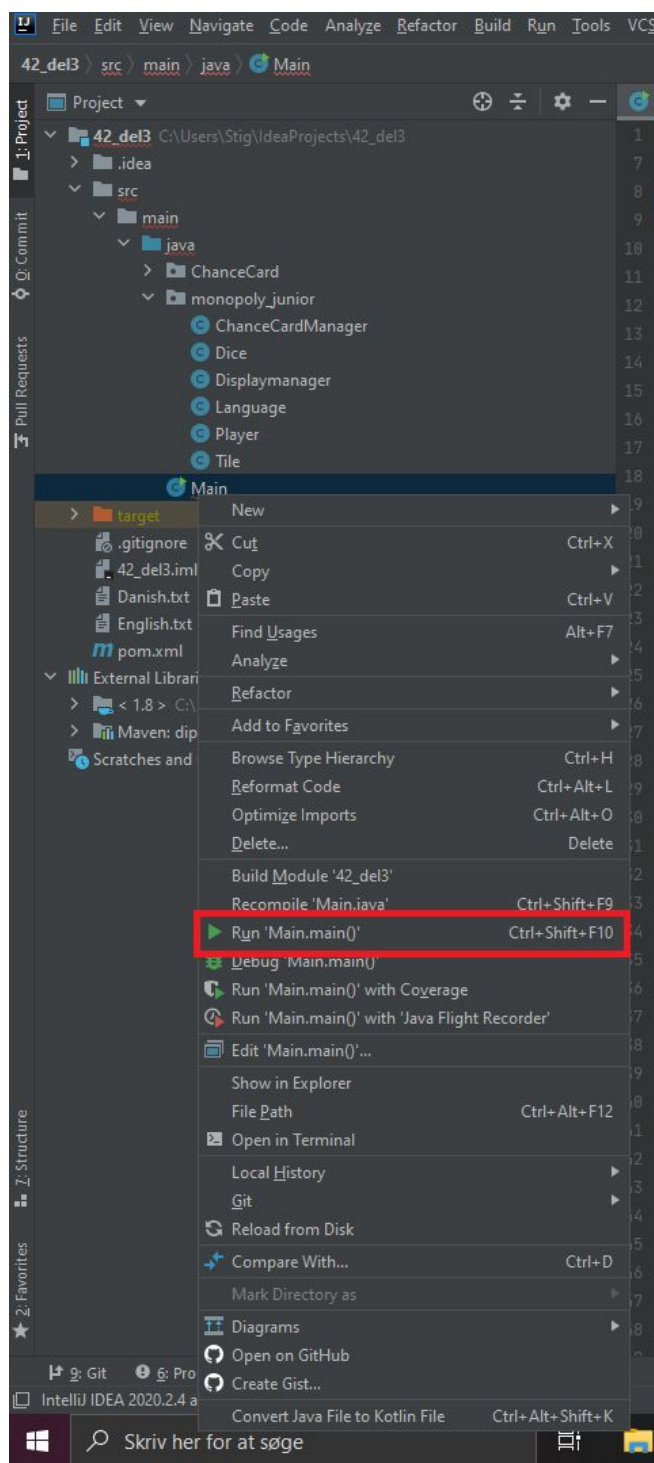


5. Hvis Junit ikke er importeret, skal man ind under test klasserne og importere Junit.
6. Hvis GUI'en ikke virker, skal man vælge "matadorgui-3.1.6.jar" og tilføje den til ens library.
7. Nu skal man gå ind og reloade projektet så man kan få Maven til at fungere. Dette skal kun gøres en gang. Der er en detaljeret tutorial længere nede som forklarer hvordan man gør.
8. Nu skal brugeren gå ned og højreklikke på main, trykke run program og køre det. Programmet starter derefter og brugeren kan begynde at spille.

Hvordan kører man programmet:

Når man er inde i IntelliJ, har hentet programmet fra github repo'en og vil gerne køre programmet skal man finde ude i outlineren den klasse som hedder "Main" og højreklikke på den:

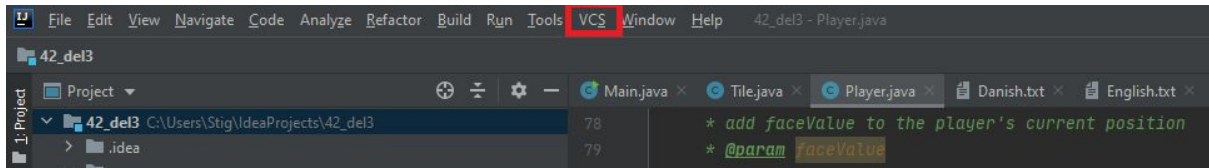




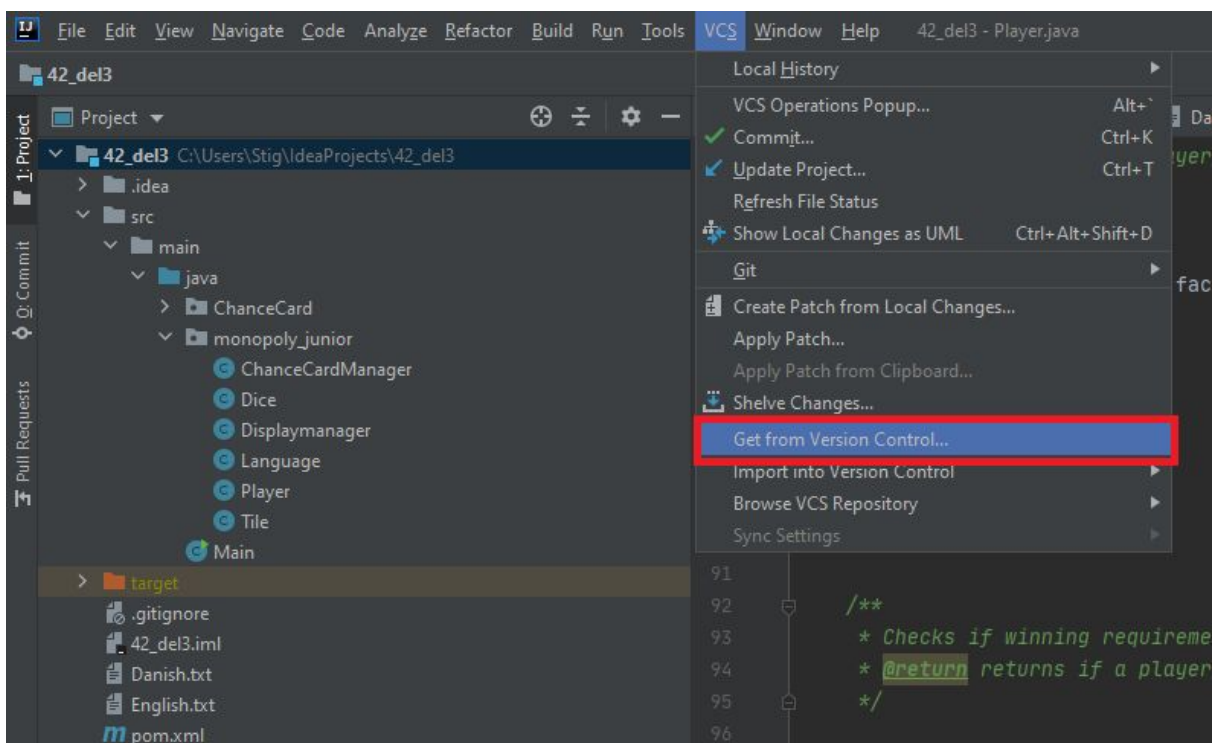
Nu skulle programmet køre og være klar til at spille.

Importeret af git repo in i IntelliJ:

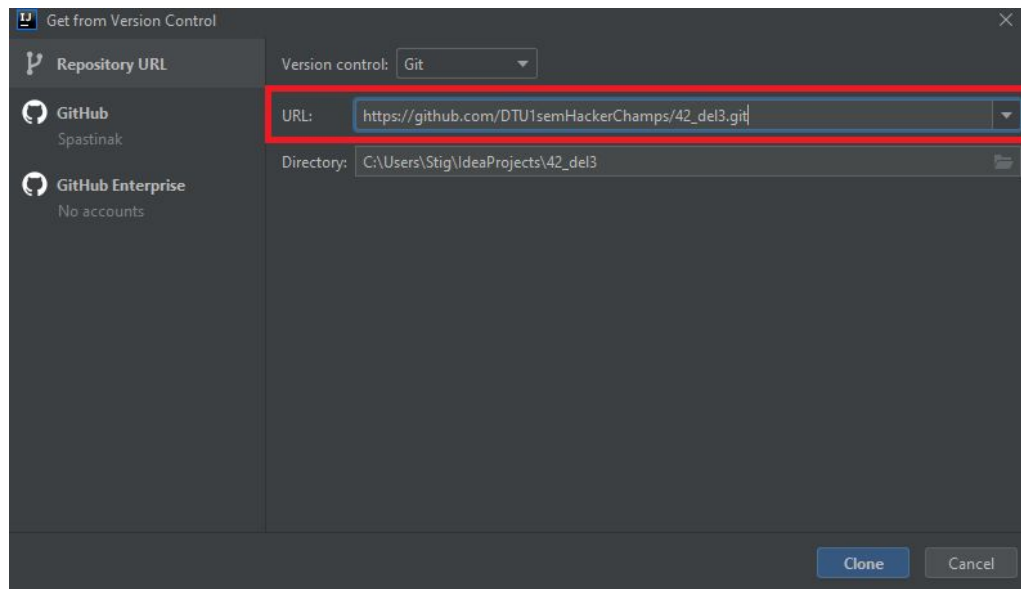
Først skal man ind i IntelliJ og finde VCS knappen øverst i programmet:



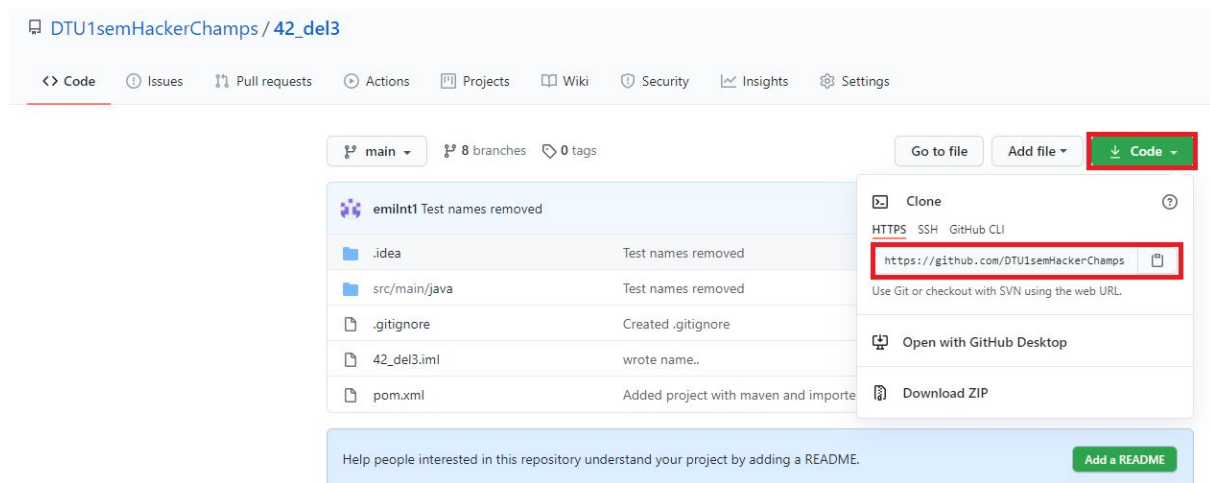
Herfra skal man finde “Get from Version Control”:



https://github.com/DTU1semHackerChamps/42_del3 og derefter trykke på Clone.



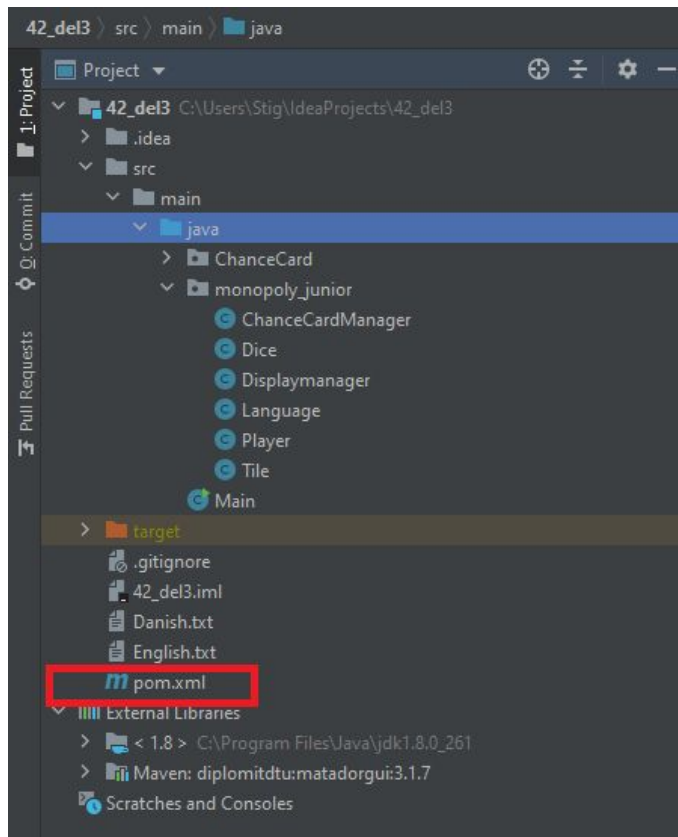
Når man er inde på github repo'et og vil finde den URL som man vil kopiere, vil der være en grøn knap som når man trykker på den vil vise den URL man skal bruge til at kopiere ind i IntelliJ:



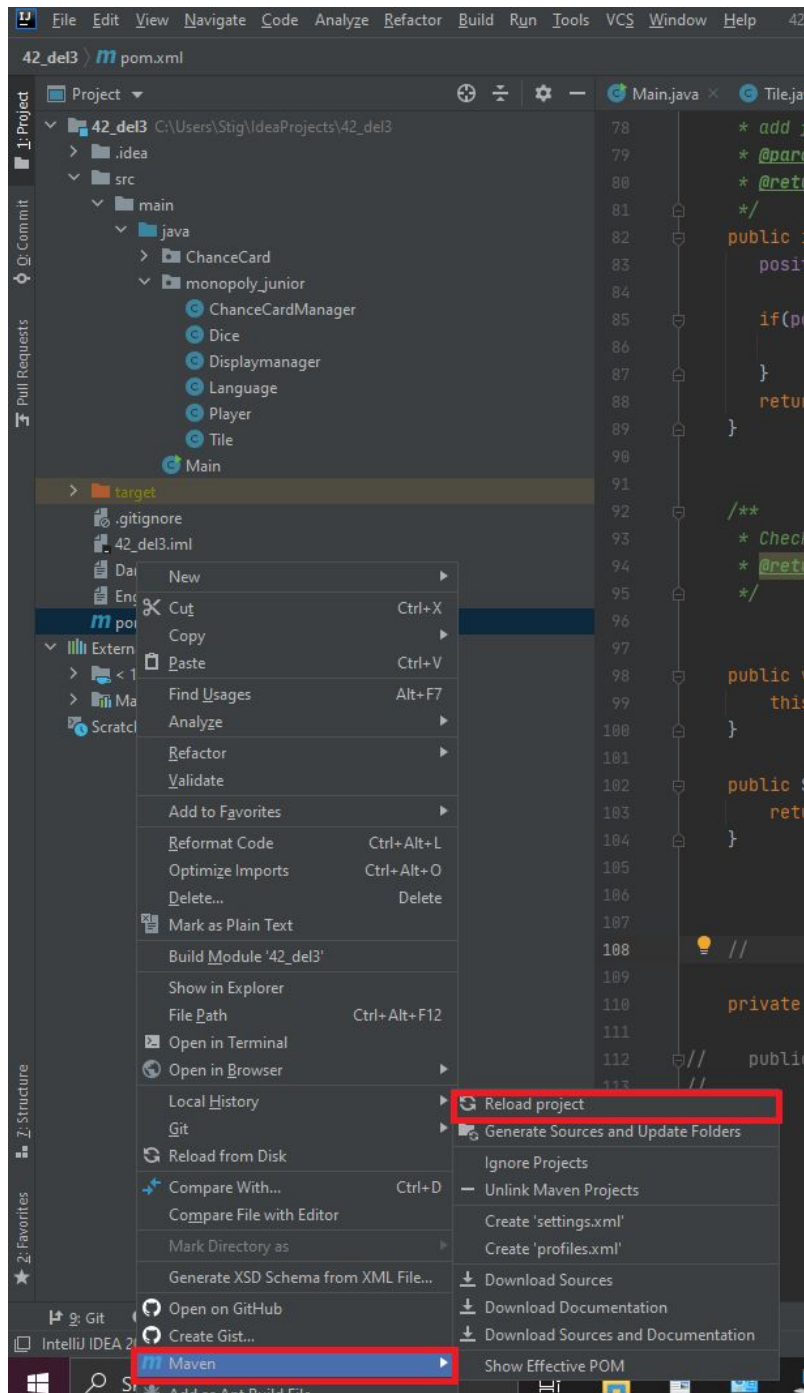
Nu skulle projektet gerne være importeret ind i IntelliJ.

Reload af pom.xml fil:

Der kan være brug for at reloade pom.xml når man først har loaded projektet ind fra github. For at reloade pom.xml filen skal man finde pom.xml i outlineren og højreklikke på den:



Dernæst skal man finde Maven, og holde over den så et nyt folder popper ud. derfra skal man klikke på “Reload Project”:



Så skulle den fil være reloadet.

Konklusion

Vi har formået at lave et program lignende til Monopoly junior, der vi kan se at det ikke er et fuldt fyldestgørende Monopoly junior da det har krævet at undlade forskellige elementer for at få en størst udførlighed for projektet. Spillet er heller ikke bug fri. Fx. når man har afsluttet et spil og der er fundet en vinder, og gerne vil genstarte spillet. Når man så genstarter spillet åbner det et nyt game vindue. Dette er et memory leak, som vi har desværre ikke fundet en løsning til.

Vi har også en mindre bug med vores animation af spiller brikkerne, som kun virker til første tur for hver af spillerne, dette skulle ikke have en større effekt end det æstetiske.

Gruppen har brugt GUI'en fra undervisningen. Ud fra test af programmet kan det konkluderes at gruppens program virker og det kan spilles af alle, store som små.

Bilag

Feltliste:

1. Start 2 (når man passerer)	
2. Burgerbaren	1
3. Pizzariaet	1
4. Chance	
5. Slikbutikken	1
6. Iskiosken	1
7. Fængsel (besøg)	
8. Museet	2
9. Biblioteket	2
10. Chance	
11. Skaterparken	2
12. Swimmingpoolen	2
13. Parkering	
14. Spillehallen	3
15. Biografen	3
16. Chance	
17. Legetøjsbutikken	3
18. Dyrebutikken	3
19. Fængsel	
20. Bowlinghallen	4
21. ZOO	4
22. Chance	
23. Vandlandet	5
24. Strandpromenaden	5

Chancekortliste:

1. Giv dette kort til BILEN, og tag et chancekort mere. BIL: På din næste tur skal du drøne frem til et hvilket som helst ledigt felt og købe det. Hvis der ikke er nogen ledige felter, skal du købe et fra en anden spiller.
2. Ryk frem til START og modtag M2.
3. Ryk op til 5 felter frem.
4. Gratis felt: Ryk frem til et orange felt. Hvis det er ledigt, får du det gratis. Ellers skal du betale leje til ejeren.
5. Ryk 1 felt frem, eller tag et chancekort mere.
6. Giv dette kort til SKIBET, og tag et chancekort mere. SKIB: På din næste tur skal du sejle frem til et hvilket som helst ledigt felt og købe det. Hvis der ikke er nogen ledige felter, skal du købe et fra en anden spiller.
7. Du har spist for meget slik. Betal M2 til banken.
8. Gratis felt: Ryk frem til et orange eller grønt felt. Hvis det er ledigt, får du det gratis. Ellers skal du betale lejen til ejeren.
9. Gratis felt: Ryk frem til et lyseblåt felt. Hvis det er ledigt, får du det gratis. Ellers skal du betale lejen til ejeren.
10. Du løslades uden omkostninger, behold dette kort indtil du får brug for det.
11. Ryk frem til Strandpromenaden.
12. Giv dette kort til katten og tag et chancekort mere. KAT: På din næste tur skal du liste dig hen på et hvilket som helst ledigt felt og købe det. Hvis der ikke er nogen ledige felter, skal du købe et fra en anden spiller.
13. Giv dette kort til HUNDEN, og tag et chancekort mere. HUND: På din næste tur skal du hoppe hen på et hvilket som helst ledigt felt og købe det. Hvis der ikke er nogen ledige felter, skal du købe et fra en anden spiller.
14. Det er din fødselsdag. Alle giver dig M1. Tillykke med fødselsdagen.
15. Gratis felt: Ryk frem til et pink eller mørkeblåt felt. Hvis det er ledigt, får du det gratis. Ellers skal du betale lejen til ejeren.
16. Du har lavet alle dine lektier, modtag M2 fra banken.
17. Gratis felt: Ryk frem til et rødt felt. Hvis det er ledigt, får du det gratis. Ellers skal du betale leje til ejeren.

18. Gratis felt: Ryk frem til skateparken for at lave det perfekte grind! Hvis ingen ejer den, får du den gratis. Eller skal du betale leje til ejeren.
19. Gratis felt: Ryk frem til et lyseblåt eller rødt felt. Hvis det er ledigt, får du det gratis, ellers skal du betale lejen til ejeren.
20. Gratis felt: Ryk frem til et brunt eller gult felt. Hvis det er ledigt, får du det gratis, eller skal du betale lejen til ejeren.

Andre kort:

1. Skibet: Alle mand om bord! Skibet er berejst og ved præcis, hvor det skal sejle hen for at holde sig forrest i spillet. Spil som skibet, der har fuld fart fremad, og sejl sejren sikkert hjem.
2. Bilen: Klar... Parat... KØR! Den hurtige og frygtløse bil skrider altid rundt i svingene og drøner videre til næste felt. Denne risikovillige racer vil gøre alt for at holde dig i front.
3. Hunden: Nuttet, legesyg og loyal - hunden er fuld af energi og altid sulten. Tag en tur rundt i Monopoly-byen med hunden. Denne lille, beslutsomme vuffer vil gøre alt for at vinde.
4. Katten: Et spændende valg. Den kløgtige kat lander altid på poterne. Katten er hurtig og adræt, og den kender hver en gade i byen. Spil som den drillesyge kat, så kan du snart liste dig

Diagrammer og modeller:

Designklassediagram:

<https://drive.google.com/file/d/1J8qnNspF321LNA9tJqCpzev6uCtp0j0J/view?usp=sharing>

Domænemodel:

<https://drive.google.com/file/d/1Ua4FJA6-ud4EZ5r9jf-fFRy2nMk3uulA/view?usp=sharing>

Usecase diagram:

https://drive.google.com/file/d/1z72iQwUe4cvd-HaHHm98m0lpLckVI2_z/view?usp=sharing

Sekvensdiagram:

<https://drive.google.com/file/d/1BVX5oqcqU60zP0tIJx2dhDx17eEge-a/view?usp=sharing>

Litteraturliste

CDIO 1

https://docs.google.com/document/d/1-XGd8bKliHMJeK0ILX8EHQ7FTbhvrboE7ldwqu_zQbQ/view

Nyborg, Mads; (2. okt 2020) Tavle noter fra indledende programmering:

https://docs.google.com/document/d/1nhgljX_WCB6znKg1sxO3x5EeXiGf-2HLcwrOqbmebHs/edit#heading=h.xnrqmx5f0nlv

Nyborg, Mads; (2. okt 2020) GUI:

<https://drive.google.com/drive/folders/0B1qlt-Xd6kaQZTdVb0hWdEt2eWM>