

CDIO3

Gruppe 16

10. november 2017



Mathias
Fager
s175182



Milishia
Moardi
s175193



Nicki
Christiansen
s170208



Semi
Seitovski
s175181



Simon
Pedersen
s175195



Thyge
S. Steffensen
s175176

DTU



Timeregnskab

Timeregnskabet kan ses via linket [her](#) eller i billag XX.

Deltager	Total timer per deltager
Mathias Fager	3
Milishia Moradi	3,5
Nicki Christiansen	10
Semi Seitovski	6
Simon Steen Pedersen	5,5
Thyge Steffensen	14,5
Total	42,5

Figur 1: Overblik over Timeregnskab den 21/11/2017

$$= \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \int_a^b \epsilon \Theta^{\sqrt{17}} + \int \delta e^{i\pi} = -1$$

$$\infty = \{2.7182818284\} \chi^2 \sum \gg \approx \lambda$$

$$\text{φ φ ε ρ τ υ θ ι ο π σ ξ}$$

1	Analyse	3
1.1	Kravliste	3
1.2	UseCases	3
1.3	UseCase diagram	7
1.4	GRASP	7
2	Design	9
2.1	Klasse diagram	9
2.2	Sekvensdiagram	10
2.3	System sekvensdiagram	10
2.4	Domænemodel	10
3	Dokumentation	11
3.1	Forklar hvad arv er	11
3.2	Forklar hvad abstract betyder	11
3.3	Fortæl hvad det hedder hvis alle fieldklasserne har en landOn- Field metode der gør noget forskelligt	11
3.4	Dokumentation for test med screenshots	11
3.5	Dokumentation for overholdt GRASP	11

$$\sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \int_a^b \epsilon \Theta + \sqrt{17} \int \delta e^{i\pi} = -1$$

UseCase Section: Opsætning af spil	Comment
Scope	Monopoly spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interesser	Spillerne er interesserede i at kunne vælge antal spillere og deres brikker
Forudsætninger	Spillet er startet op, og spillerne har nu mulighed for at vælge antal spillere og ønskede brikker
Success garanti	Der er blevet valgt antallet af spillere, og hver spiller har valgt sin brik, herefter er spillet klar til at blive spillet

$$\begin{aligned} & \Delta \int_a^b \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = -1 \\ & = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \\ & \quad \infty = \{2.7182818284\} \\ & \quad \chi^2 \sum \gg \approx \lambda \end{aligned}$$

Fully dressed UseCase:

UseCase Section: Spillerne slår med terningerne	Comment
Scope	Monopoly spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interesser	Spillerne er interesseret i at kunne trykke på en knap, og få et billede af to terninger med tilfældige værdier
Forudsætninger	Spillet er startet op, og spillerne har valgt antallet spillere og deres ønskede brikker
Success garanti	Der er blevet valgt antallet af spillere, og hver spiller har valgt sit navn, herefter er spillet klar til at blive spillet
Hoved succes scenarie	Spillerne får udgivet en værdi af to terninger, og lander derefter på et felt
Alternative udfald	Negative udfald: - IOOuterActive har opdateret spillet, og derved opstår der en fejl når spillerne slå med terningerne, der kan ende i at der ikke bliver slået to terninger - Systemet blokerer for en spillers tur - En spiller hopper fra/på, og derved skal spillet startes om
Specielle krav	- Enheden som spillet kører på skal være kompatibel med Java - Spillerne skal kunne interagere med GUI'en ved brug af mus eller touch - Der skal være plads på enheden til at kunne hente spillet
Hyppighed	Hver tur bliver der slået med terninger

UseCase Section: Spiller køber et felt	Comment
Scope	Monopoly spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interesser	Spillerne er interesseret i at købe det aktuelle felt
Forudsætninger	Spillet er i gang og en spiller har slået med terningerne
Success garanti	Spilleren køber og ejer nu feltet

$$= \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \quad \Delta \int_a^b \epsilon \Theta + \Omega \int \delta e^{i\pi} = -1$$

$$\chi^2 \sum \gg \approx \lambda$$

$$\{2.7182818284\} \circ \partial \phi \epsilon \pi \theta \iota \omega \pi \sigma \xi$$

UseCase Section: Spiller lander på et eget felt	Comment
Scope	Monopoly spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interesser	****
Forudsætninger	Spillet er i gang og en spiller lander på et felt som er ejet af en anden spiller
Success garanti	En spiller lander på et felt der er ejet af en anden spiller og får derfor en negativ effekt

UseCase Section: En spiller taber	Comment
Scope	Monopoly spil af IOOuterActive
Level	User-goal
Primær Aktør	Terningerne
Stakeholder og interesser	Spillerne er interesseret i at deres balance ikke når 0, og dermed taber
Forudsætninger	Spillerne har slået med terningerne
Success garanti	En spiller lander på 0, og er ude af spillet

UseCase Section: Spillet afsluttes	Comment
Scope	Monopoly spil af IOOuterActive
Level	User-goal
Primær Aktør	IOOuterActive
Stakeholder og interesser	IOOuterActive er interesseret i at programmet viser en vinder og afsluttes
Forudsætninger	Alle spillere undtagen en, har fået en balance på 0
Success garanti	Spillet viser en vinder og kan derefter afsluttes

$$= \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) = \int_a^b \epsilon \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = -1$$

$$\chi^2 \sum \gg \approx \lambda$$

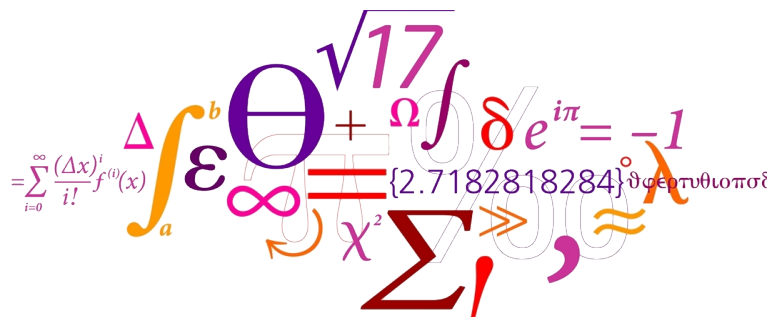
$$\{2.7182818284\}$$

$$\partial \phi \partial \tau \partial \theta \partial \sigma \partial \xi$$

$$\sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \int_a^b \varepsilon \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = -1$$

3. Low coupling
4. Controller
5. High cohesion
6. Indirection
7. Polymorphism
8. Protected variations
9. Pure fabrication

(Der skrives mere når vi er nået længere i projektet)



Herunder ses en række 'design steps', som skal hjælpe os med at lave Monopoly Junior spillet.

```

classDiagram
    class GUI
    class GameController {
        <<Superklasse>>
        Package: FatClass
        - Players : String[]
        Responsibilities:
        - GameFlow
    }
    class Player {
        <<Stereotype>>
        Package: Player
        - playerName : String
        - playerColor : int
        - playerLocation : int
        - playerPiece : int
        + getPlayerName() : String
        + getPlayerColor() : int
        + getPlayerLocation() : int
        + getPlayerPiece() : int
        + setPlayerName(String name) : void
        + setPlayerColor(int id) : void
        + setPlayerLocation(int loc) : void
        + setPlayerPiece(int id) : void
        Responsibilities:
        - Indeholder og administrerer playerattributter
    }
    class Dice {
        <<Stereotype>>
        Package: Dice
        - int dice1
        - int dice2
        + roll() : void
        + getDice1() : int
        + getDice2() : int
        + getSum() : int
        Responsibilities:
        - Værdier af øjnene
    }
    class Wallet {
        <<Stereotype>>
        Package: Player
        - balance
        + setBalance(int balance) : void
        + changeBalance(int chBal) : void
        + getBalance() : int
        Responsibilities:
        - Holde styr på spillernes balance
    }
    class assignFieldEffect {
        <<Stereotype>>
        Package: Player
        - propertyID : int
        Responsibilities:
        - Bestemme hvad der skal ske*
        - Har ID på alle grundene
    }
    class specialFields {
        <<Stereotype>>
        Package: Player
        -specialField[] : int
        + getSpecialField(int fieldID) : void
        Responsibilities:
        - Håndterer specialegrunde
        - Start
        - Fængsel
        - osv
    }
    class propertyField {
        <<Stereotype>>
        Package: Player
        Responsibilities:
        - Håndterer frie og solgte grunde
    }
    class chanceCardField {
        <<Stereotype>>
        Package: Player
        -chanceCard[] : int
        +getChanceCard() : void
        Responsibilities:
        - Håndterer chance kort
    }

    GUI --> GameController
    GameController --> Player : adminAttri 1 to 2.6
    GameController --> Dice : rollDice 1 to 1
    GameController --> assignFieldEffect : 1 to 01
    GameController --> Wallet : adminMoney 1 to 1
    assignFieldEffect --> specialFields : 1 to 1
    assignFieldEffect --> propertyField : 1 to 1
    assignFieldEffect --> chanceCardField : 1 to 1
  
```

The diagram illustrates the structure of a game system. At the top, a **GUI** class interacts with the **GameController** class. The **GameController** class, marked as a **«Superklasse»** in the **Package: FatClass**, contains a **Players : String[]** attribute and a **Responsibilities: - GameFlow**. It has directed associations to several other classes: **Player** (labeled **adminAttri** with multiplicity 1 to 2.6), **Dice** (labeled **rollDice** with multiplicity 1 to 1), **assignFieldEffect** (multiplicity 1 to 01), and **Wallet** (labeled **adminMoney** with multiplicity 1 to 1). The **Player** class, a **«Stereotype»** in **Package: Player**, has attributes **playerName : String**, **playerColor : int**, **playerLocation : int**, and **playerPiece : int**, along with methods **getPlayerName() : String**, **getPlayerColor() : int**, **getPlayerLocation() : int**, and **getPlayerPiece() : int**. Its responsibility is **Indeholder og administrerer playerattributter**. The **Dice** class, also a **«Stereotype»** in **Package: Dice**, has attributes **int dice1** and **int dice2**, methods **roll() : void**, **getDice1() : int**, **getDice2() : int**, and **getSum() : int**, with the responsibility **Værdier af øjnene**. The **Wallet** class, a **«Stereotype»** in **Package: Player**, has a **balance** attribute and methods **setBalance(int balance) : void**, **changeBalance(int chBal) : void**, and **getBalance() : int**, with the responsibility **Holde styr på spillernes balance**. The **assignFieldEffect** class, a **«Stereotype»** in **Package: Player**, has a **propertyID : int** attribute and responsibilities **Bestemme hvad der skal ske*** and **Har ID på alle grundene**. It has directed associations to **specialFields**, **propertyField**, and **chanceCardField**, all with multiplicity 1 to 1. The **specialFields** class, a **«Stereotype»** in **Package: Player**, has a **-specialField[] : int** attribute and a **getSpecialField(int fieldID) : void** method, with the responsibility **Håndterer specialegrunde** and a list of **Start**, **Fængsel**, and **osv**. The **propertyField** class, a **«Stereotype»** in **Package: Player**, has the responsibility **Håndterer frie og solgte grunde**. The **chanceCardField** class, a **«Stereotype»** in **Package: Player**, has a **-chanceCard[] : int** attribute and a **getChanceCard() : void** method, with the responsibility **Håndterer chance kort**.

Klasse diagrammet bygger på vores umiddelbare overvejelser, såvel som vores use case's. Dette er for at illustrere sammenspillet mellem vores klasser og deres associationer. Dog skal det siges, at dette er en skitse og den aktuelle programmering kan variere heraf.

$$\sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \int_a^b \epsilon \Theta + \Omega \int \delta e^{i\pi} = -1$$

2.2 Sekvensdiagram

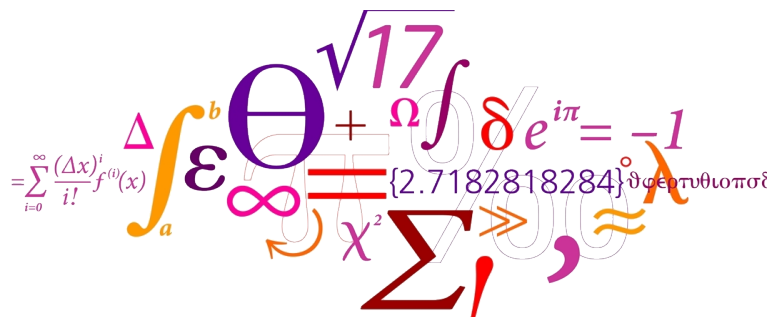
Vi har her lavet et sekvensdiagram, der skal skabe et overblik over hvordan aktøren, her spilleren, kommunikerer med spillet.

2.3 System sekvensdiagram

Vi har her lavet et systemsekvensdiagram for at forhøje gennemsigtigheden ved bruge af 'chanceCard' klassen.

2.4 Domænemodel

Ved hjælp af domænemodellen vil vi trække paralleller mellem den virkelige verden og programmeringen. Domænemodellen er en visuel repræsentation af konceptklasser og 'objekter fra den virkelige verden'. Ved hjælp af denne kan vi også oplyse kunden om, hvad vi vil lave.



3 Dokumentation

3.1 Forklar hvad arv er

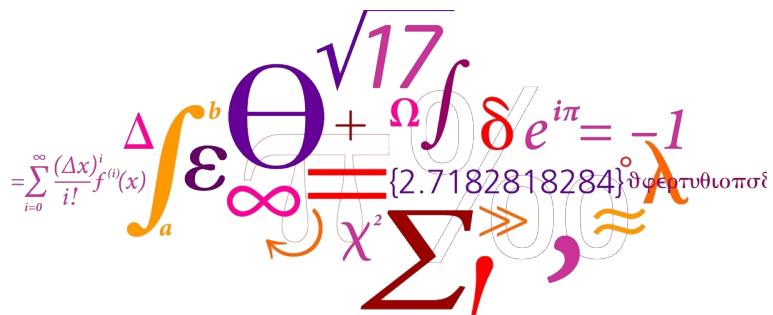
3.2 Forklar hvad abstract betyder

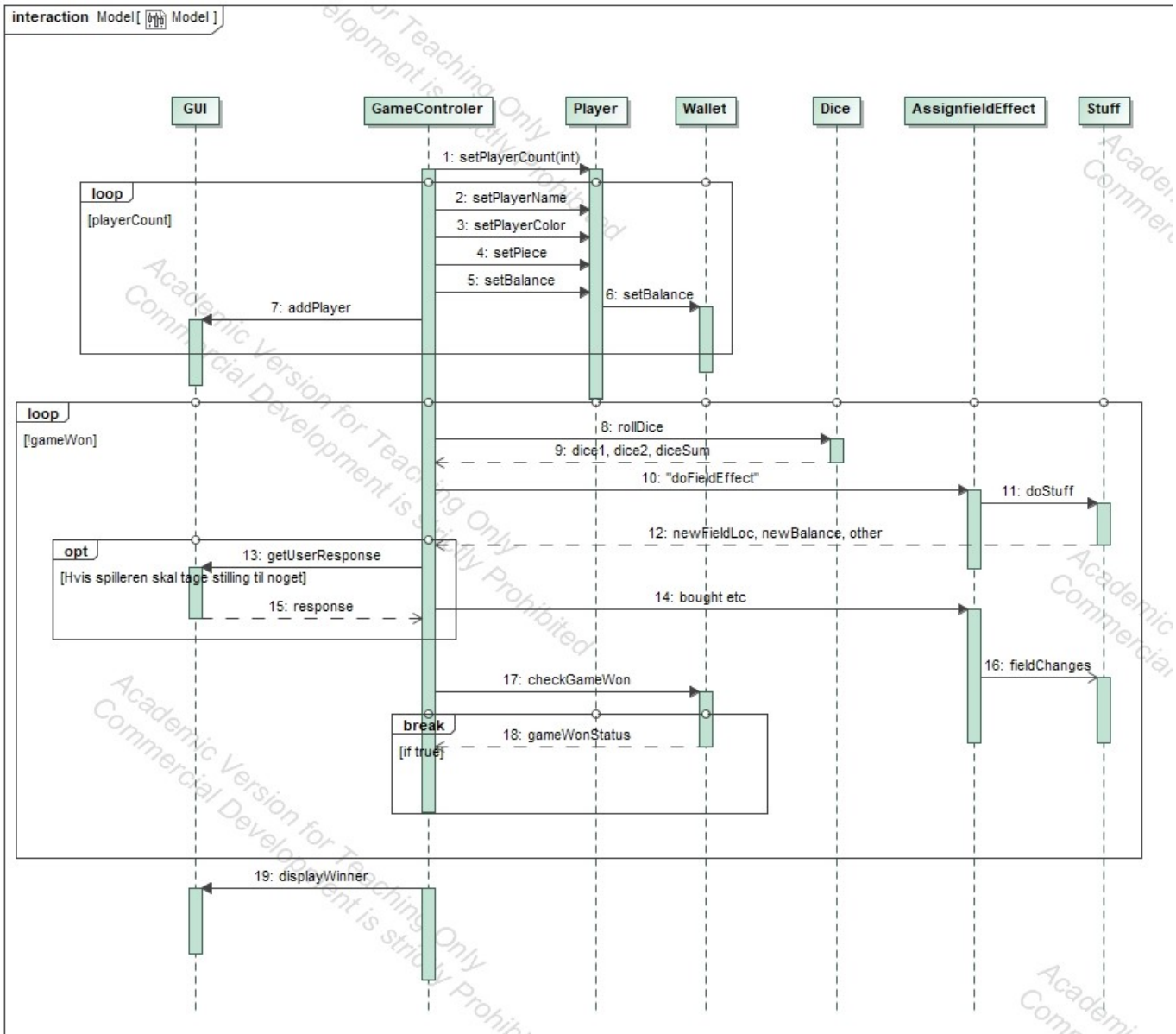
3.3 Fortæl hvad det hedder hvis alle fieldklasserne har en `landOnField` metode der gør noget forskelligt

Hvis alle fieldklasserne gør brug af den samme `landOnField` metode er det fordi denne metode er en super metode nedarvet til de forskellige subklasser, dette gør at alle klasserne kan gøre brug af samme metode. Eksempelvis, hvis vi har en masse dyr som klasser, kat, hund, kanin etc. så kan de alle nedarve super metoden `eat()` fra superklassen som hedder `SurvivalRequirements`, eftersom disse er metoder alle dyrene får brug for, så vil det give mening at lave det til en superklasse med supermetoder , så der holdes lav kobling og høj kohæsion, samt undgås kopiering af kode og høj mulighed for genbrug.

3.4 Dokumentation for test med screenshots

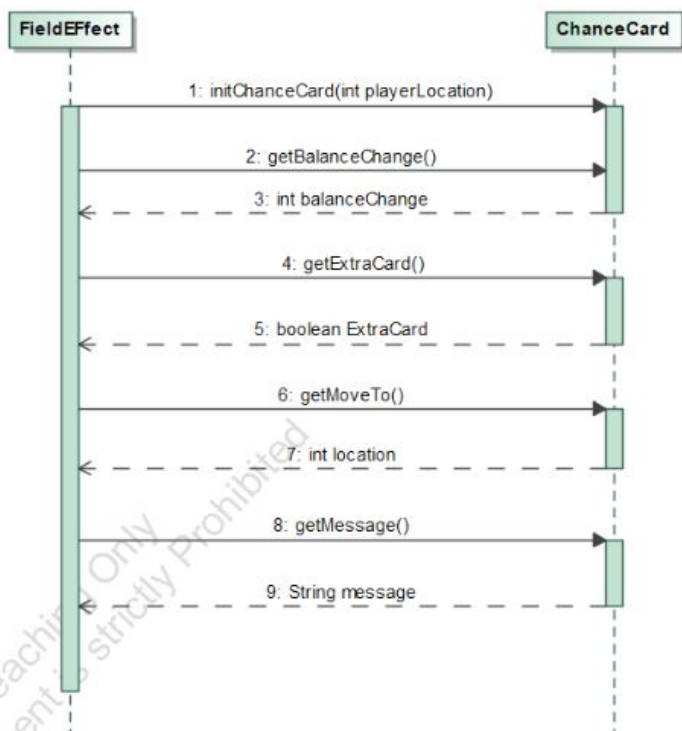
3.5 Dokumentation for overholdt GRASP





Figur 4: Sekvensdiagram tegnet i MagicDraw

$$\begin{aligned}
 &= \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \\
 &\int_a^b \epsilon \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = -1 \\
 &= \{2.7182818284\} \lambda \\
 &\chi^2 \sum_i \gg \approx
 \end{aligned}$$



initChanceCard
 - initialisere et chancekort: Vælger et kort nummer (1ud af 18) og skal benytte den hele vejen igennem denne session.

balanceChange
 - Både en positiv og negativ værdi, der skal lægges til balancen

extraCard
 - Hvis spilleren skal trække et chance kort til, er denne positiv

moveTo
 - Giver en int som skal lægges til spillerens lokation
 - BEMÆRK: Denne tager ikke højde for antal felter på pladen, dette skal gøres i AssignField

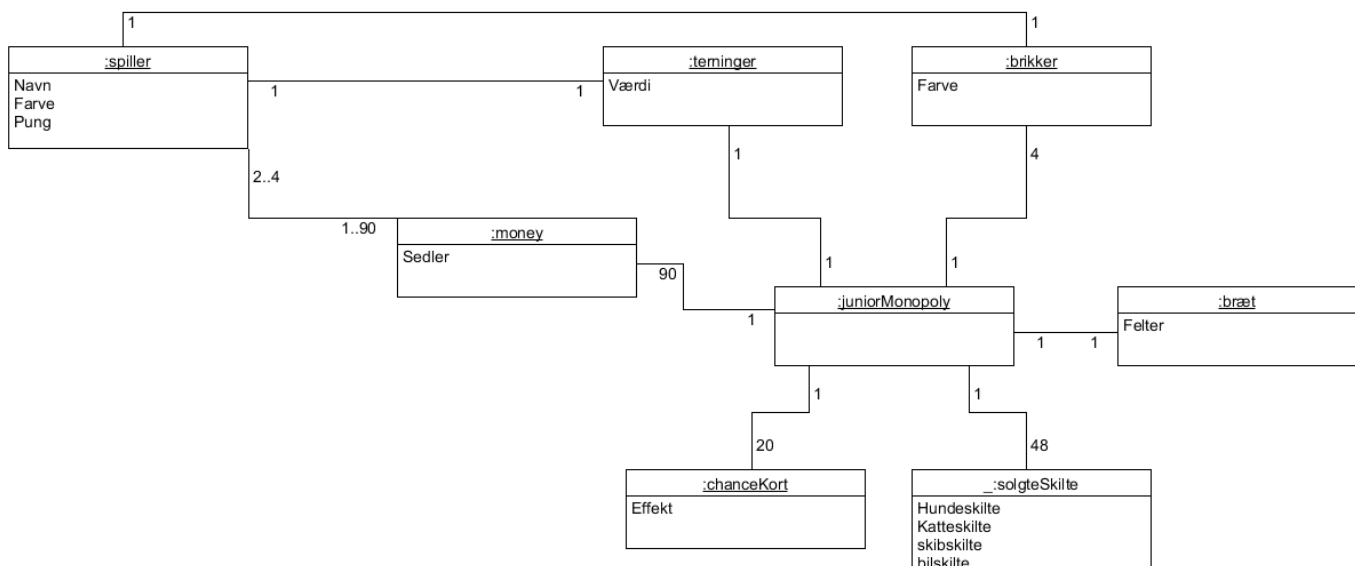
message
 - indeholder besked, som skal vises til brugeren.

Init
 - set message
 - set moveTo
 - set extraCard
 - set balance

Figur 5: Systemsekvensdiagram tegnet i MagicDraw

$$= \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) \quad \Delta \int_a^b \epsilon \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = -1$$

$$\infty = \{2.7182818284\} \quad \chi^2 \sum \gg \approx \lambda$$



Figur 6: Domænemodel tegnet i UMLet

$$= \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x) = \int_a^b \epsilon \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = -1$$

$\infty = \{2.7182818284\}$
 $\frac{1}{\chi^2} \sum_{i=1}^{\infty} \frac{1}{i!} \approx \lambda$