3-ugers projektopgave, vinteren 2015

02312 Indledende programmering

Projektnavn: 11_final
Emne: Matadorspil
Afleveringsfrist: Mandag 19/01-2015 12:00
CDIO gruppenummer: 11

Gruppemedlemmer (studienummer, efternavn, fornavn):

s144860, Sondrup, Mathias

s123157, Adamsen, Frederik

s144846, Eriksen, Robert

s144845, Ørnby, Victor

s144844, Thomsen, Mads

s144858, Hansen, Nicklas

Rapporten er afleveret elektronisk via Campusnet og der skrives derfor ikke under.
Rapporten indeholder 37 sider inkl. denne side og bilag.

Code repository URL: https://github.com/adamsen9/CDIO-Final/commits/master

Timeregnskab

Time regnskab	Ver: 14-01-2014						
		Opgaver					
Dato	Deltager	Analyse og design	Implementation	Test	Dokumentation	Andet	I alt
05-01-2015	Nicklas	3					3
05-01-2015	Frederik	3				1	4
05-01-2015	Mathias	3					3
05-01-2015	Victor	3					3
05-01-2015	Robert	3					3
05-01-2015	Mads	3					3
06-01-2015	Nicklas	2	3				5
06-01-2015	Mathias		4				4
06-01-2015	Mads	1				1	2
06-01-2015	Victor		4				4
06-01-2015	Robert		4				4
06-01-2015	Frederik				1,5		1,5
07-01-2015	Nicklas		4				4
07-01-2015	Frederik		6				6
07-01-2015	Mathias		3				3
08-01-2015	Frederik		3				3
08-01-2015	Mads		4				4
08-01-2015	Nicklas		3	1			4
08-01-2015	Victor		2		2		4
08-01-2015	Robert		3				3
08-01-2015	Mathias		5				5
09-01-2015	Nicklas		2		3		5
09-01-2015	Victor		2		2		4
09-01-2015	Mathias		3				3
09-01-2015	Robert		3				3
09-01-2015	Mads		3				3
12-01-2015	Nicklas		2	2			4
12-01-2015	Mads				1		1
14-01-2015	Nicklas		1	2	2	1	6
14-01-2015	Mads		1	2	2	1	6
14-01-2015	Mathias		1	2	2	1	6
14-01-2015	Frederik		1	2	2	1	6
14-01-2015	Robert		1	2	2	1	6
14-01-2015	Victor		1	2	2	1	6
15-01-2015	Nicklas		1		3	2	6
15-01-2015	Mads						0
15-01-2015	Mathias				3		3

15-01-2015	Frederik			1	1		2
15-01-2015	Robert	4		2			6
15-01-2015	Victor			2			2
17-01-2015	Nicklas					1	1
18-01-2015	Nicklas				3	2	5
18-01-2015	Mathias				2	2	4
19-01-2015	Nicklas				1	1	2
19-01-2015	Mathias		0,1	2	1	1	4,1
19-01-2015	Frederik			2	2	1	5
19-01-2015	Victor		1	2		1	4
		Total anvendt	178,6				

Forord

Rapporten er skrevet af seks studerende ved DTU, der går på første semester af diplomingeniør Softwareteknologi. Projektperioden forløb fra d. 05/01-2015 til d. 19/01-2015.

I forbindelse med studiet blev der stillet en opgave i form af et projekt hvor der skulle fremstilles et matador-lignende spil. Kort sagt skulle man kunne være 2-6 spillere og kunne rykke rundt på 40 felter.

Rapporten er tilblevet som dokumentation for projektforløbet og for udviklingen af koden bag spillet. Målgruppen for rapporten er eksaminator og censor - rapporten og projektet skal bruges som grundlæggende faktor til eksamen.

Tak til vejlederne og hjælpelærere af projektet:

Hjælpelærer(e):

Ronni Dalsgaard
Daniel Kolditz Rubin-Grøn

Vejleder:

Stig Høgh

Indholdsfortegnelse

Contents

Indledning

Projektoplægget var i bund og grund at der skulle udvikles et Matador spil. Dette indebærer altså at man skal kunne være et antal spillere der kan slå med en terning og lande på nogle felter. Disse felter kan være af forskellig type og alt efter type kan de enten købes eller have en anden funktion. Projektoplægget er vedlagt som bilag nr. 1.

Under projektforløbet er der blevet brugt forskellige hjælpemidler til at assistere og støtte tilblivelsen af CDIO Final.

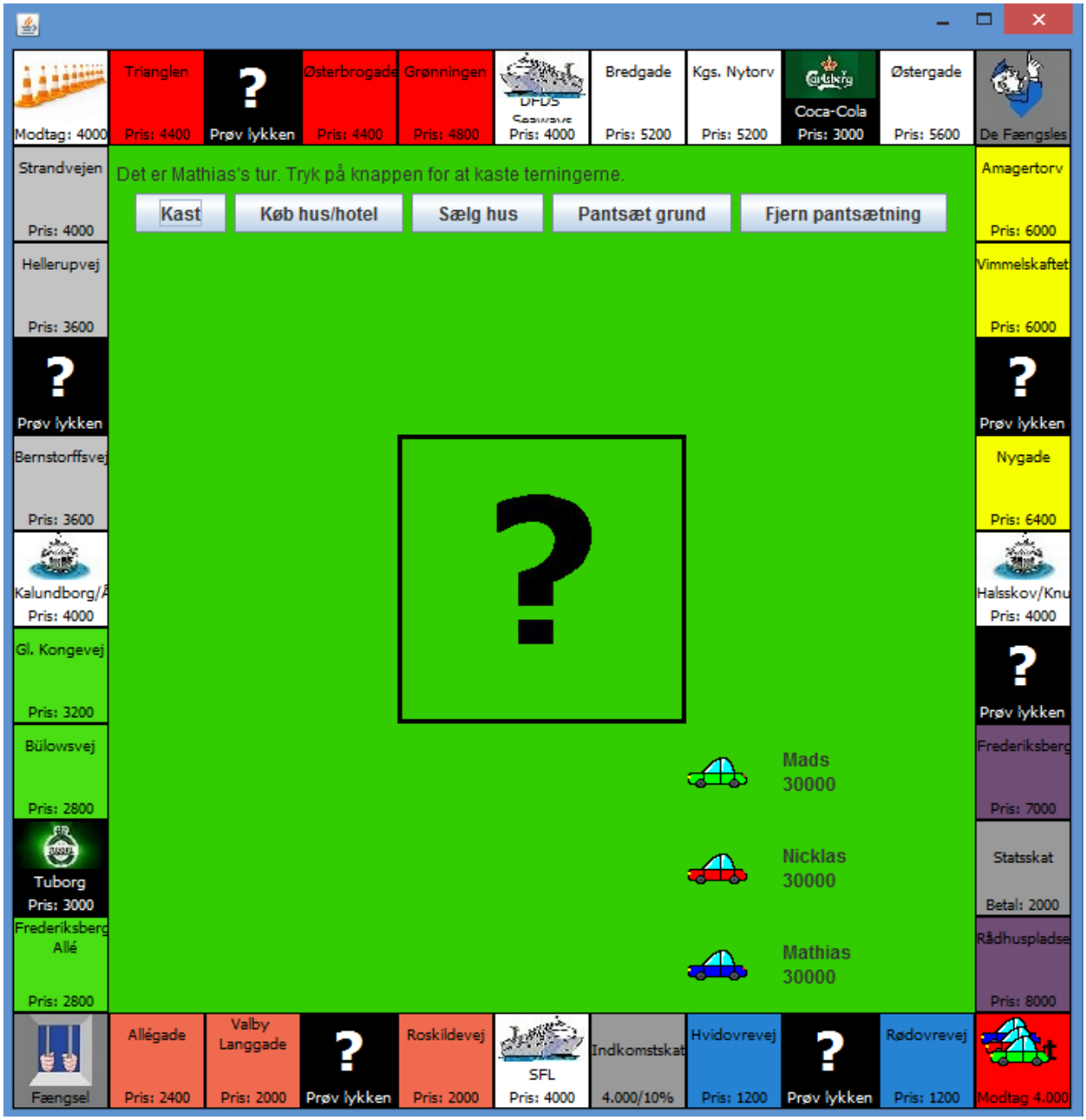
På det organisatoriske plan har hjemmesiden "GroupRoom" hjulpet gruppen til at få et overblik samt kategorisere de forskellige arbejdsopgaver, samt ansvarsdeling blandt gruppemedlemmerne.

Google Drive har gennem projektforløbet været base for alle dokumenter. Google Docs har udmærkede egenskaber for et projektforløb da redigering af dokumenter kan foretages af flere personer samtidig, og eventuelle ændringer bliver vist med det samme for de andre brugere.

Til at lave diverse UML-diagrammer er der brugt MagicDraw og til versionering er der brugt Git, herunder GitHub. Der vil blive kommet nærmere ind på versionering senere i rapporten.

Indledningsvis vises, på figur 1 forneden, et billede af det udviklede produkt. Grunden til at dette vises så tidligt i rapporten, er for allerede fra start af at give læseren en idé om, hvad det egentlig er, projektet drejer sig om.

Figur 1: Det udviklede spil



Foroven ses som sagt Figur 1, der viser vores spil. Det kan ses at vi har en spilleplade med 40 felter hvor alle de traditionelle felter fra et dansk Matador-spil er med og funktionelle. Vi har opdelt de forskellige grunde i farve-kategorier og vi har fået lavet langt de fleste “Prøv Lykken”-kort. Der er dog enkelte typer af kort vi ikke har fået implementeret, dette er beskrevet i afsnittet “Begrænsninger/Fravalg af funktioner”.

Analyse

Kravspecifikationer

Vi har i denne opgave ikke haft en "kunde" som har stillet os nogle krav der skal opfyldes men vi har selv udarbejdet en lang række krav, som er nødvendige for at Matador-spillet fungerer efter hensigten.

Funktionelle krav

1. Spilleren skal kunne lande på et felt.

2. Spilleren skal slå med to terninger og rykke det samlede antal øjne frem, venstre om på pladen.

3. Der skal være mellem 2 og 6 spillere.

4. Hver spiller skal starte med en pengebeholdning på 30.000.

5. Spillet skal afsluttes når alle på nær én spiller er bankerot.

6. Der skal være 40 felter på spillepladen.

7. Der skal findes 8 typer felter: Street, Parking, Tax, Brewery, Shipping, Jail, Chance og Start.

    7.1. En Street skal kunne købes, og lander man på et allerede ejet Street skal man betale en afgift til ejeren. Prisen varierer alt efter hvilken Street man er landet på.

    7.2. Lander man på et Parking får man udbetalt en bonus på 4000.

    7.3. På Tax trækkes penge fra den spiller der er landet på feltet. Spilleren skal vælge om denne vil betale et fast beløb eller en procentdel af spillerens pengebeholdning.

    7.4. Brewery er et felt man skal kunne købe, og lander en spiller på en ejet Brewery, skal spilleren betale penge. Formlen for beløbet der skal betales afhænger af, hvor mange Brewery der ejes. Hvis en spiller ejer én Brewery betales, slår den landende spiller med terningerne og betaler det slåede antal øjne x 100 x 1 (antal af ejede Brewery).

    7.5. Shipping er et felt der skal kunne købes. Lander man på et allerede ejet felt, skal man betale penge til ejeren, medmindre grunden er pansat.

7.5.1. Lejen der skal betales ændrer sig alt efter hvor mange shipping felter ejeren er i besiddelse af.

7.5.1.1. Leje hvis 1 shipping felt ejes: 500
7.5.1.2. Leje hvis 2 shipping felter ejes: 1000
7.5.1.3. Leje hvis 3 shipping felter ejes: 2000
7.5.1.4. Leje hvis 4 shipping felter ejes: 4000

7.5.2. Man kan maksimalt eje 4 shipping felter på én gang.

7.6. Lander en spiller på feltet “De fængsles”, skal man gå direkte i fængsel og får altså ikke 4.000 kr. selv om man passerer “Start”. Man kan komme ud af fængslet på flere måder.

7.6.1. Ved at betale en bøde på 1.000 kr., inden man kaster.

7.6.2. Ved at kaste 2 ens antal øjne; man flytter da straks det antal pladser, øjnene viser, og får et ekstrakast.

7.6.3. Man kan ikke blive i fængsel mere end tre omgange. Får man ikke to af samme antal øjne, når man kaster tredje gang, må man betale bøden, 1.000 kr. og flytte, som øjnene viser.

7.6.4. Den spiller der er fængslet har stadig ret til at købe grunde der går på auktion.

7.6.5. Lander en spiller på feltet “Fængsel” er man imidlertid blot på besøg og fortsætter næste gang uden straf.

7.7. Lander en spiller på et “Chance” felt, skal denne trække et kort og følge anvisningerne på kortet.

7.8. Hver gang en spiller kommer til, eller passerer, “Start”, modtager denne 4.000 kr. fra banken.

8. Det skal være muligt for en spiller at bygge huse på en grund spilleren ejer.

8.1. Det skal være muligt at bygge op til 4 huse på en grund og derefter et hotel.

8.2. Idet at 4 huse er bygget på en grund skal det være muligt for en spiller at bygge den sidste opgradering, et hotel.

8.3. Antallet af huse på en grund bestemmer den leje en anden spiller skal betale for at lande på grunden.

8.4. Det skal være muligt at sælge et hus eller hotel tilbage til banken.

8.4.1. Sælges en bygning får ejeren præcis halvdelen af det bygningen oprindeligt kostede tilbage.

9. Det skal være muligt at pantsætte grunde man selv ejer.

9.1. Det er kun muligt at pantsætte en grund idet, det er spillerens egen tur.

9.2. Idet en grund pantsættes får man halvdelen af grundens købsværdi udbetalt af banken.

9.3. Det er muligt for ejeren af grunden at købe en pantsat grund tilbage for det samme beløb han fik udbetalt idet han pantsatte den.

9.4. Idet en spiller lander på en pantsat grund betaler han ikke leje til ejeren af grunden.

10. Når en spiller lander på en grund der ikke er ejet, som vedkommende ikke vil eller kan købe selv, går dette skøde på auktion.

10.1. Auktion er en proces hvor alle spillere har mulighed for at byde på en grund.

10.2. Idet en auktion starter skal alle spillere byde på en grund.

10.3. En spiller kan vælge ikke at ville deltage i auktionen.

10.4. Idet en spiller går bankerot går alle hans grunde på auktion.

11. Man taber når man skal betale penge til enten banken eller spiller og ikke har penge nok.

11.1. Idet man taber bliver alle grunde man ejer sat på auktion, og det er muligt for alle spillere at byde på en given grund.

Ikke funktionelle krav

1. Projektet skal bygge videre på CDIO del 1, CDIO del 2 og CDIO del 3.

2. Programmet skal kunne køres på computerne i DTU's databarer.

3. Programmet skal være udviklet i Java.

Begrænsninger / fravalg af funktioner

Undervejs i projektet har vi stort set lavet et fuldt funktionelt Matador-spil. Der er dog enkelte funktioner vi ikke har implementeret. Dette er et aktivt valg vi har taget da vi mener at spillet fungerede fint uden.

Dette gælder blandt andet chance kortet "Matador-legatet", hvor man kan få en økonomisk støtte hvis man er blanket helt af.

"Benådelse Kortet" er ikke implementeret.
"Oliepriserne stiger" er ikke implementeret.
"Det er din fødselsdag, du modtager 200 af modspillerne" er ikke implementeret.

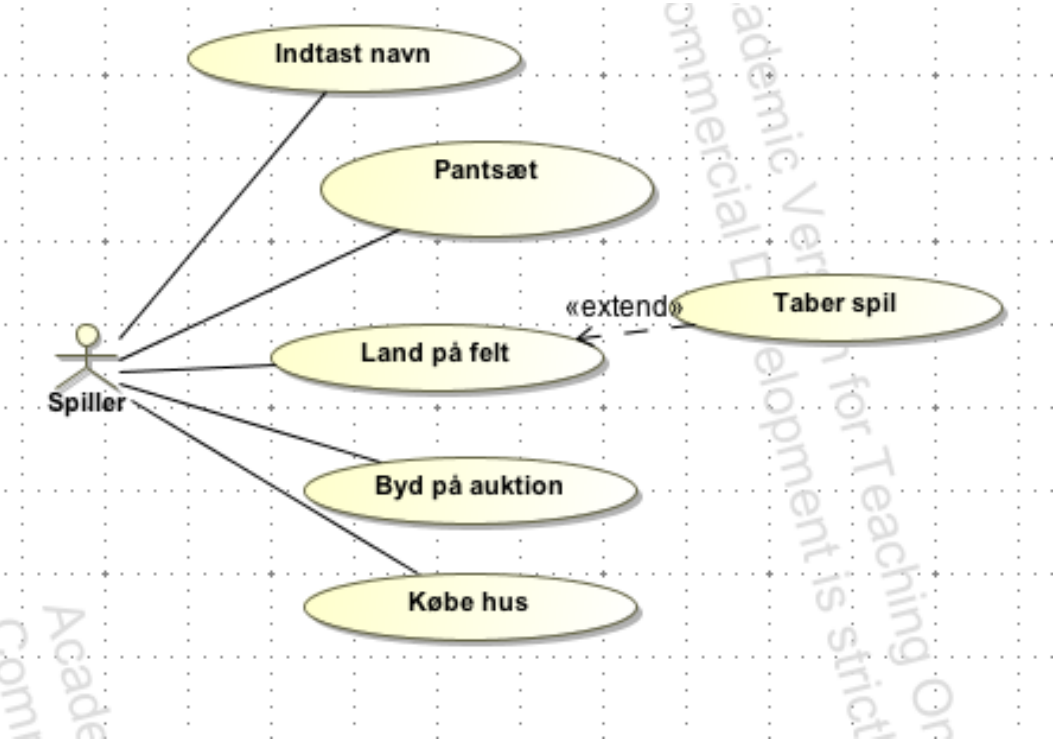
"Dobbelt pris for leje, når en spiller ejer alle felter i en kulør" er ikke implementeret.

"Man kan ikke købe grunde i første omgang", dette mener vi ikke var nødvendigt at implementere så man *ikke* kunne købe på første omgang. Det er rigtigt at den spiller, der starter, i teorien har en mulighed for at komme foran men da det er terninger man slår med, er det jo helt tilfældigt hvad der sker. Den første spiller kan således sagtens blive overhalet og desuden kan det også ske, at den første spiller lander på et felt der har en negativ effekt på ham og derfor giver dette også en balance i spillet.

Use-case diagram

Nedenfor, figur 2, ses vores usecase diagram.

Figur 2: Use-case diagram



Som det kan ses på diagrammet ovenfor er der specificeret 6 use cases, hvor den ene af disse extender en af de andre. Disse er beskrevet nedenfor.

Use case beskrivelser

Som det kan ses på diagrammet ovenfor, Figur 2, så består use case diagrammet blot af fem use cases, disse er alle beskrevet nedenfor.

Use Case	Land på Felt
ID	1
Kort beskrivelse	Spiller lander på et felt
Hovedaktører	Spiller
Sekundære Aktører	Ingen
Forhåndsbetingelser	Spillet er startet. Spiller har kastet terningerne.
Hovedforløb	1. Spiller trykker “Kast” 2. Terningerne kastes

	3. Spilleren rykker antal øjne frem. <Alternativt forløb afhængigt af hvad spilleren lander på>
Resultat	Spilleren er rykket til et nyt felt
Alternative Forløb	<Parkering> Spiller modtager 500 eller 5000 valuta. <Tax> Spiller skal på det ene tax felt betale 2000 i skat af sin totalbalance. På det andet tax felt kan spiller vælge mellem at betale 10% skat af sin totalværdier eller 4000 penge. <Street> Der forekommer 3 scenarier for Spiller ved at lande på Street. Enten skal Spiller betale leje, såfremt en modspiller ejer feltet. Hvis feltet er ledigt, kan spiller købe feltet såfremt spiller har penge nok. Spiller har også muligheden for at ignorere feltet. <Shipping> De samme 3 scenarier fra Street gør sig gældende for Fleet. Prisen for leje er dog anderledes, da den er fastsat efter hvor mange Fleet felter ejeren ejer. <Brewery> De samme 3 scenarier fra Street gør sig gældende i Brewery. Prisen for leje erdog anderledess. Lejen er i Brewery bestemt af, hvad Spiller slår med et nyt par terninger, efter at have landet på feltet. <Jail> Spilleren skal på det ene jail felt ryge i fængsel. Spilleren kan først komme ud af fængslet ved enten at betale 1000 eller slå 2 ens ved kast af terning. På det andet jail, er spilleren blot på besøg i fængslet og det udløser ingen effekt at lande på feltet. <Chance>

	Spilleren trækker tilfældigt et chancekort med instrukser som udføres automatisk af spillet selv

Use Case	Indtast navn
ID	2
Kort beskrivelse	Spilleren indtaster navn
Hovedaktører	Spiller
Sekundære Aktører	Ingen
Forhåndsbetingelser	Spillet er startet
Hovedforløb	1. Spiller indtaster navn 2. Navnet godkendes ellers <alternative forløb>
Resultat	Spilleren er sendt videre til kast med terninger
Alternative Forløb	1. Spillerens navn er ikke blevet godkendt 2. Spilleren bliver spurgt om et nyt navn. 3. Dette bliver ved indtil spilleren indtaster et gyldigt navn.

Use Case	Spiller taber spil
ID	3
Kort beskrivelse	Spillerens balance kommer under 0
Hovedaktører	Spiller
Sekundære Aktører	Ingen
Forhåndsbetingelser	Spiller har spillet og fået en balance der er mindre end nul.
Hovedforløb	1. Spiller lander på et felt der gør at dennes balance går under 0. 2. Spilleren udgår som aktør i spillet.

	3. Spillers portfolio af felter bliver frigivet så de igen kan købes af de resterende spillere.
Resultat	Spiller udgår fra spillet
Alternative Forløb	Ingen

Use Case	Købe huse
ID	4
Kort beskrivelse	Spilleren køber huse
Hovedaktører	Spiller
Sekundære Aktører	Ingen
Forhåndsbetingelser	1. Spillet ejer alle andre felter af samme kulør. 2. Spilleren har penge til at købe hus 3. Det er spillerens tur 4. <Alternativ forløb>
Hovedforløb	1. Spiller køber hus
Resultat	Spilleren opretter hus på den valgte grund. Lejen justeres derefter
Alternative Forløb	1. Spilleren har ikke penge nok og afvises i at købe felterne. 2. Spilleren ejer ikke alle felterne i den pågældende kulør, og kan derfor ikke købe huse.

Use Case	Pantsæt
ID	5
Kort beskrivelse	Spilleren pantsætter et felt
Hovedaktører	Spiller

Sekundære Aktører	Ingen
Forhåndsbetingelser	1. Spillet ejer feltet
Hovedforløb	1. Spiller pantsætter feltet
Resultat	Spilleren for den pantsatte værdi af feltet der er halvdelen af købsprisen.
Alternative Forløb	

Use Case	Byd på auktion
ID	6
Kort beskrivelse	En anden spiller har tabt spillet og de resterende spillere kan byde på de efterladte felter
Hovedaktører	Spiller
Sekundære Aktører	De andre spillere der vil deltage i auktionen.
Forhåndsbetingelser	1. En spiller har tabt spillet og de ejede felter frigives til auktion.
Hovedforløb	1. De resterende spillere kan vælge at byde på felterne. 2. De spillere der vælger at

	byde på auktionen indgår i en auktionsrunde.Hver enkelt spiller får mulighed for at afgive sit bud, og derefter hæve det hvis spilleren ønsker at auktionen for det pågældende felt skal fortsætte. 3. <Alternativ forløb>
Resultat	Spilleren der giver det højeste bud på det pågældende felt, bliver ejer af feltet
Alternative Forløb	1. Spiller vælger at udgå fra auktionen og for ikke fratrukket nogle penge.

Navneords Analyse

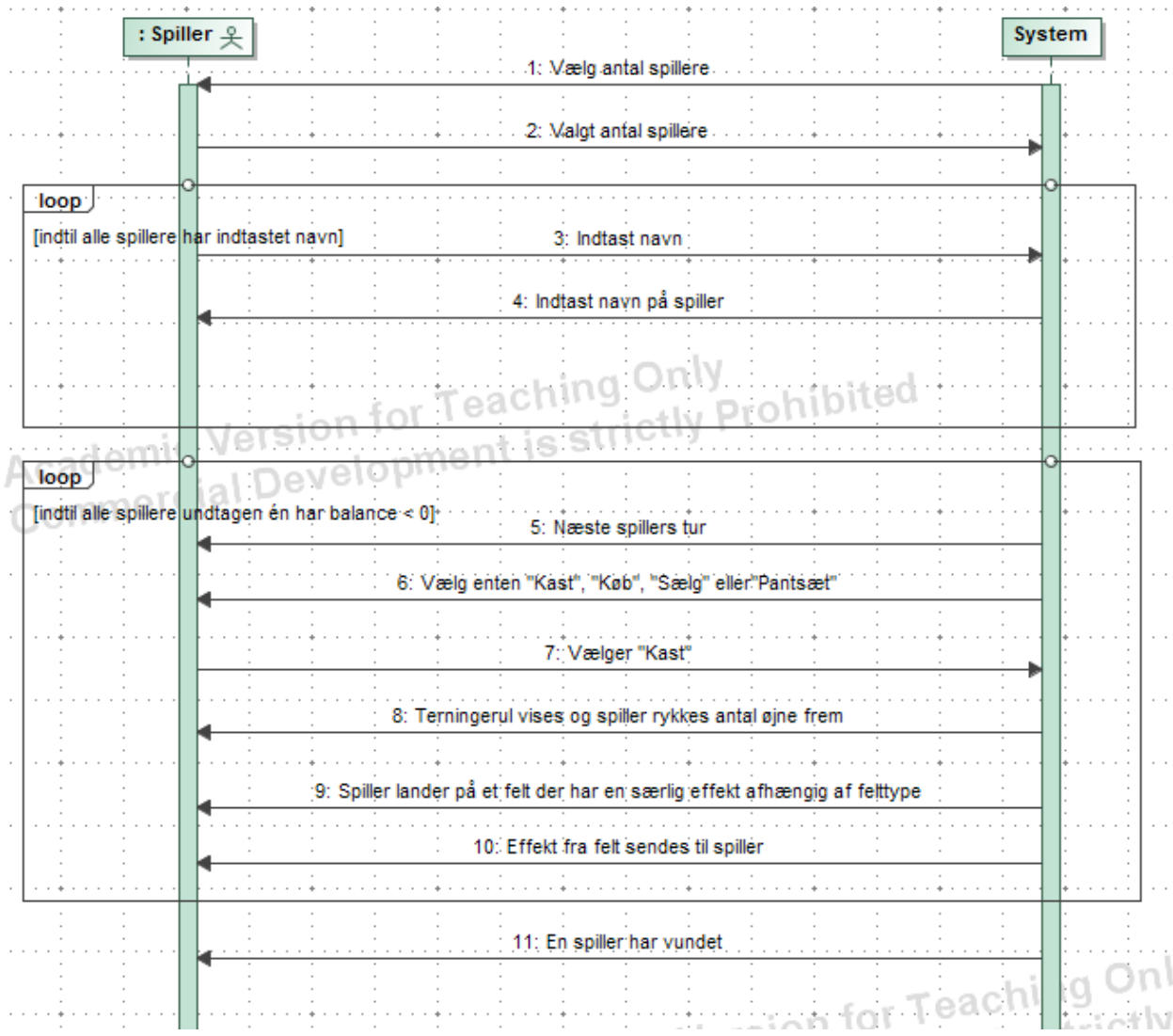
I starten af projektet udarbejdede vi en navneords analyse ved at undersøge et Matador brætspil. Vi fandt frem til følgende vigtige navneord som har været med til at give forslag til hvilke klasser vi skulle have i vores program:

- Skøde
- Spiller
- Felt
- Bank
- Auktion
- Huse
- Hoteller
- "Prøv lykken"-kort
- Fængsel
- Penge
- Leje

System Sekvens Diagram

System sekvens diagrammer viser systemets interaktion med spilleren i forskellige scenarier af usecases. I tilfældet på Figur 3 forneden, ses forløbet når et spil startes og en spiller vælger at kaste med terningerne. Vi har holdt diagrammet overordnet for at holde overblikket. Et system sekvens diagram kan meget hurtigt blive uoverskueligt hvis man tager hver mulige situation med og det er netop overblik der er meningen med disse sekvens diagrammer.

Figur 3: System Sekvens Diagram

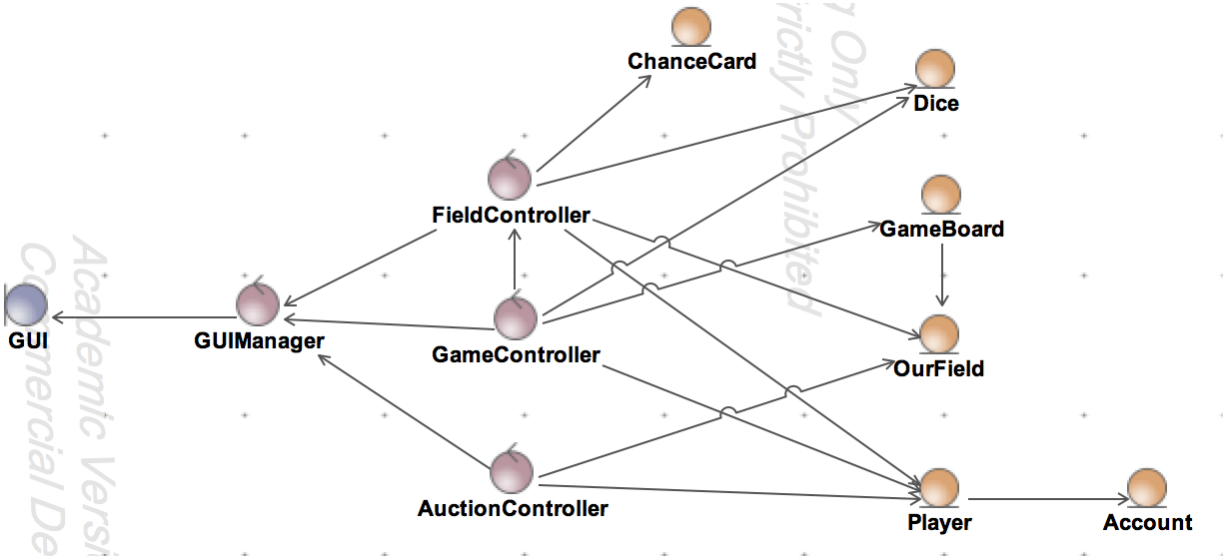


Design

BCE-model

Forneden, på Figur 4, ses vores BCE model. Den er meget simplificeret, da den ellers ville blive alt for uoverskuelig. Der findes flere FieldController men vi har valgt at vise dem som én generel Controller på modellen her, der findes også flere felt klasser end den ene der er vist her som “OurField” dog er disse ligesom de andre udgaver af “FieldController” ikke vist da det ville gøre diagrammet betydeligt mere uoverskueligt.

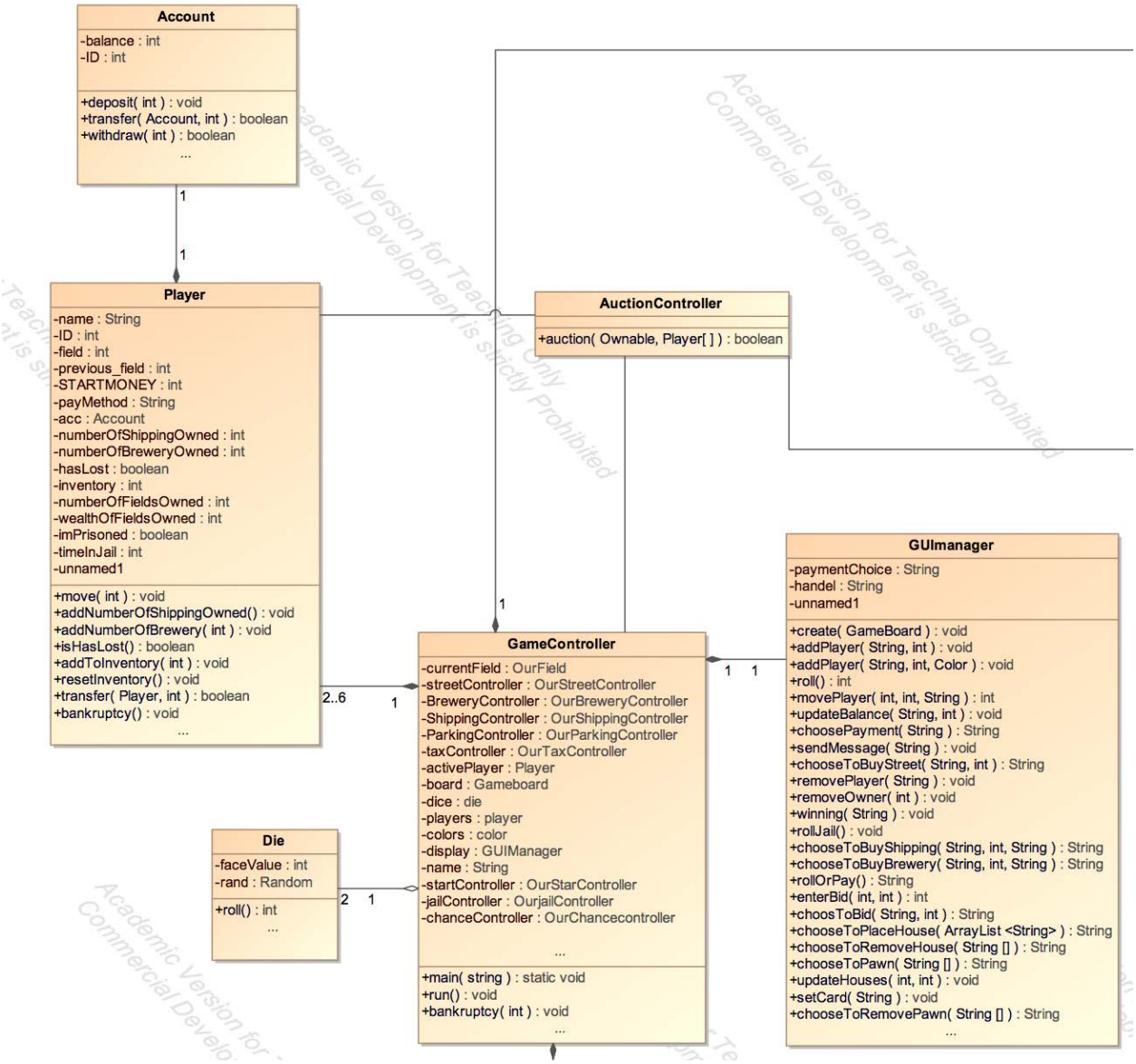
Figur 4: BCE-model



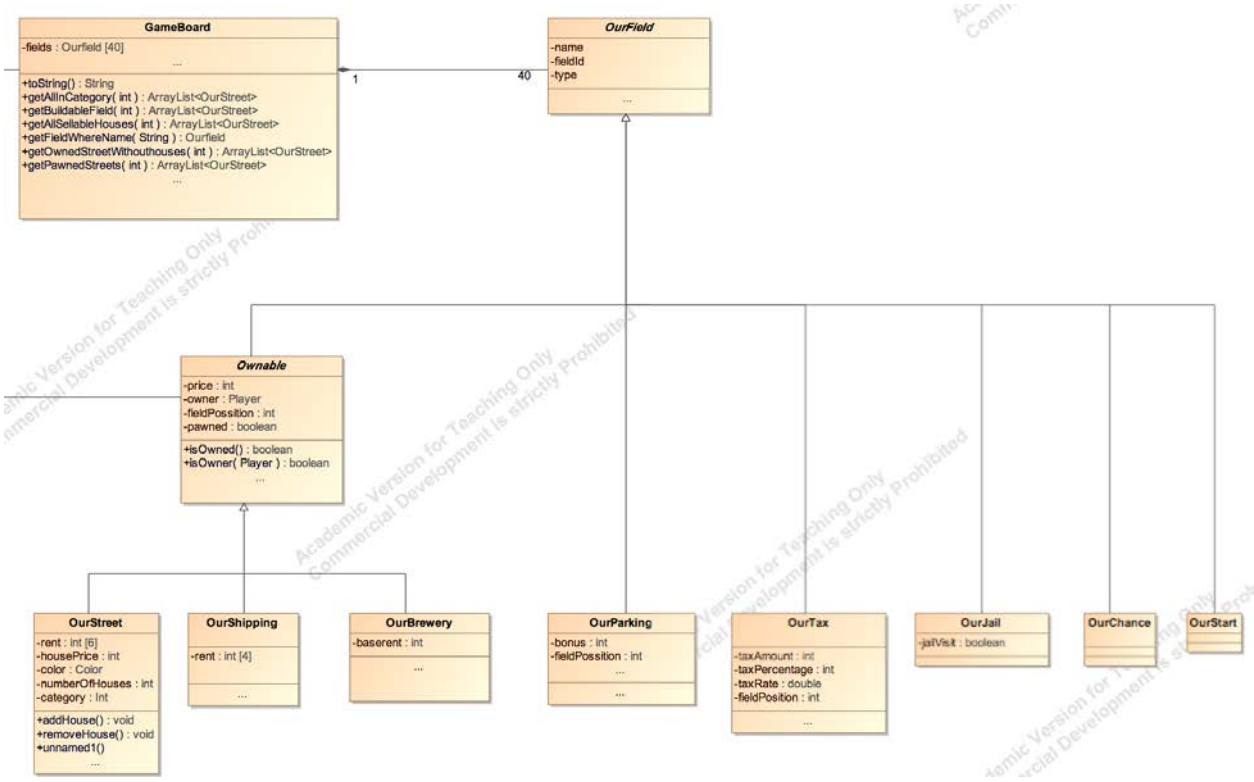
Klassediagram

Forneden ses vores klassediagram, Figur 5. Denne viser de forskellige klasser der forefindes i vores kode samt hvilke relationer de måtte have til hinanden. Diagrammet er delt op i tre bidder således at det ikke bliver for småt. Måden vi har udviklet dette spil er

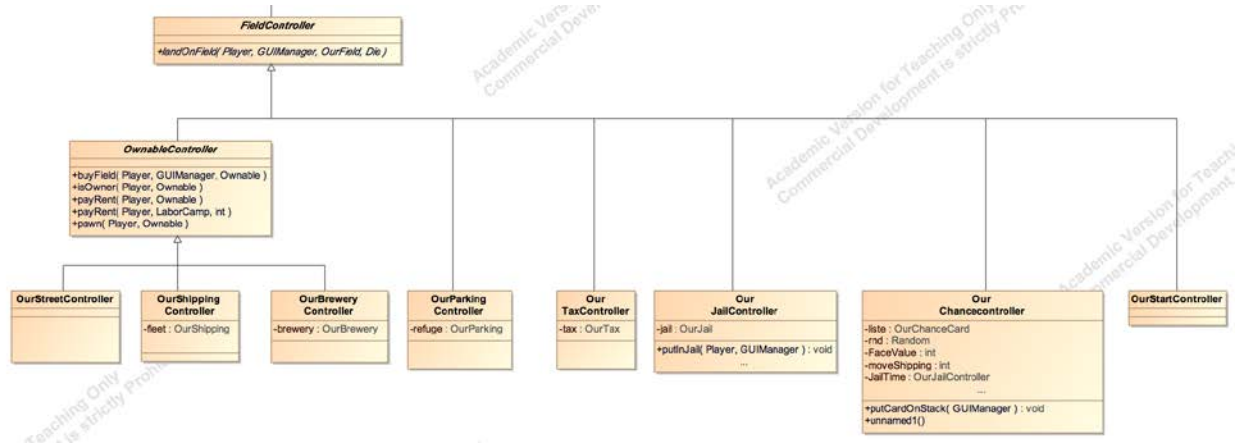
Figur 5: Klassediagram del 1



Figur 6: Klassediagram del 2



Figur 7: Klassediagram del 3

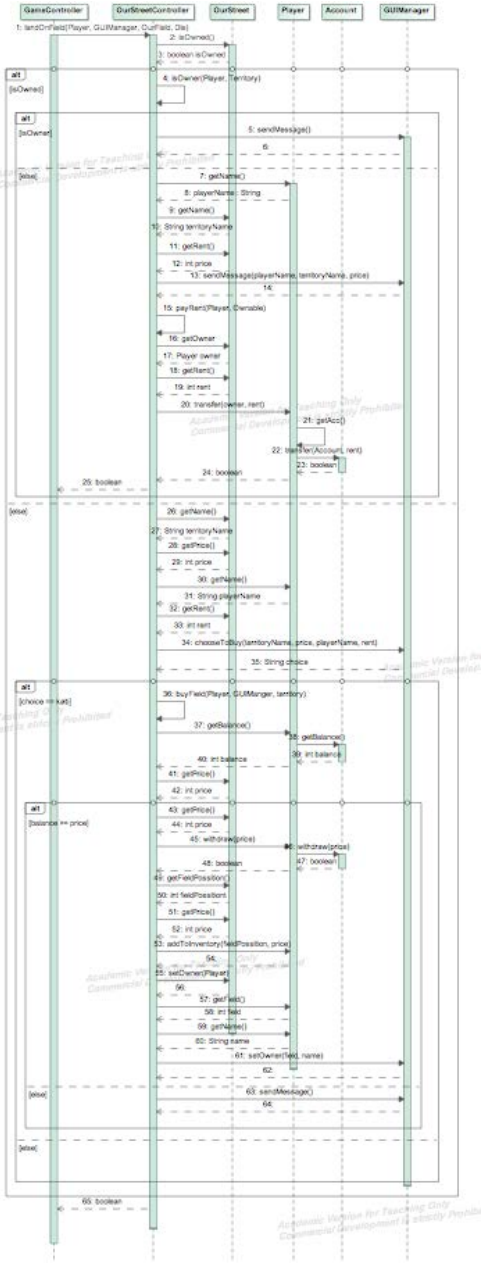


Som det kan ses på de to ovenstående dele af klassediagrammet (figur 6 og 7), så har vi i vores program to arvehierakier. Det første (figur 6) er hierarkiet af de felter der findes på pladen af matador spillet. Det andet (figur 7) er for de controllere der står står for at styre disse felter. Disse controllere svare en til en med felterne, da der findes en controller til at styre hver af felterne. Denne nedarvning gør det også muligt for at kalde “landOnField” metoden på det felt der bliver landet på uden, at skulle tjekke hvilket felt det er ved hjælp af switch/if statements. Rent programmeringsmæssigt gøres dette ved at ligge alle vores felt controllers i et array og så kalde landOnField på det nummer felt i arrayet det er der er landet på.

Design sekvensdiagram for landOnField (Street)

Nedenfor ses design sekvensdiagrammet for landOnField metoden i klassen StreetController. Det er denne metode der bliver kaldt hver gang en spiller lander på et street felt. LandOnField metoden findes i alle felt controller klasserne, den er dog implementeret forskelligt i alle disse, den nedarves fra superklassen “FieldController”.

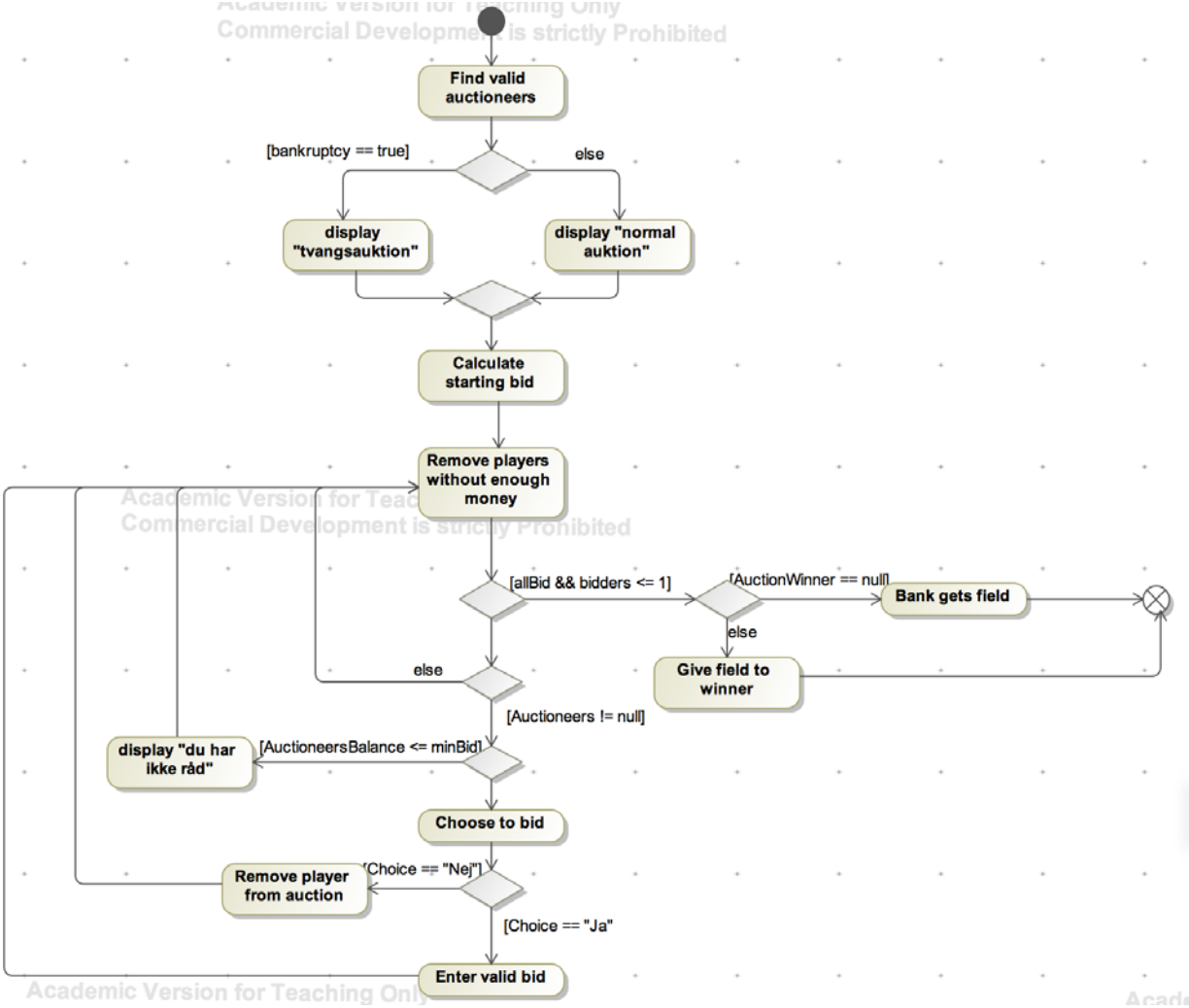
Figur 8: Design Sekvensdiagram for Street



Acitivity Diagram for auction

På figur 9 nedenfor ses at aktivitetsdiagram for auction metoden i AuctionController klassen. Formålet med denne metode er at afholde en auktion mellem de spilleren der er en del af spillet. Grunden til at udelukkende "auction" metoden af AuctionController klassen er skildret er at det udelukkende er denne metode i AuctionController klassen der indeholder kompleks logik.

Figur 9: Activity diagram for Auction



Versionsstyring

Til versionsstyring har gruppen gennem hele projektforløbet brugt et distribueret versioneringssystem kaldet Git. Git bruges til at holde styr på ændringer i koden, samt hvem der har lavet hvad. Git muliggør at man kan gå tilbage til tidligere versioner af koden, der er gemt i historikken. Git virker udmærket når man sidder og skriver kode alene og lokalt, men dets begrænsninger bliver tydelige når der er flere personer om at skrive koden. I sådanne situationer er det smart at have et centralt repository - altså et centralt "gemmested" for koden. I dette projekt er der gjort brug af den gratis online tjeneste GitHub som central repository.

Der kan naturligvis opstå problemer hvis flere personer sidder og retter i de samme filer. Dette må derefter løses ved manuel "merging" af de ændrede filer. Dette har ikke været det store problem i dette projekt da der har været en forholdsvis koordineret programmering.

Tests

Automatiske tests

Vi har valgt at foretage en række unit test, i et forsøg på at minimere chancen for fejl i vores projekt. Disse test er foretaget på Account og Player klasserne, de resterende klasser er testet ved hjælp af manuelle tests. Test cases for disse kan ses herunder.

Test af Player, foretaget i PlayerTest.java

Test Case: **TestInitialCondition**
Test Case ID: 36
Test Case beskrivelse: Formålet med denne test er, at teste at Player konstruktøren virker som vi forventet at den gør.
Forudsætninger:

- Ingen

Testprocedure:

- En spiller oprettes

Forventede resultater:

- Spillerens navn er blevet sat til det vi indtastede.
- Spillerens ID er blevet sat til det vi indtastede.
- Spillerens balance er lig startkapitalen på 30.000.
- Spilleren ejer ingen felter.

Faktiske resultater:

- Som forventet er spillerens navn sat til det vi angav.
- Som forventet er spillerens ID er blevet sat til det vi angav.
- Som forventet er spillerens balance lig startkapitalen på 30.000.
- Som forventet ejer spilleren ingen felter.

Status: Bestået

Test Case: **testAddSet**
Test Case ID: 37
Test Case beskrivelse: Formålet med denne test er, at teste de 3 add metoder som der findes i Player klassen.
Forudsætninger:

- En spiller skal være oprettet

Testprocedure:

- Der tilføjes en til mængde af shippings spilleren ejer
- Der tilføjes en til mængde af breweries spilleren ejer
- Der tilføjes et felt spil spillerens inventory

Forventede resultater:

- Spilleren ejer et shipping felt.
- Spilleren ejer en brewery.

- Spilleren ejer et felt.
- Spillerens første tal i spillerens inventory svare til det felt id som der blev tiføjet.

Faktiske resultater:
- Som forventet ejer spilleren et street
- Som forventet ejer spilleren en brewery
- Som forventet ejer spilleren et felt
- Som forventet svare det første tal i spillerens inventory til det felts id som der blev tilføjet.

Status: Bestået

Test Case: **testLost**
Test Case ID: 38
Test Case beskrivelse: Formålet med denne test er at teste at spilleren starter med ikke at have tabt, samt at vi kan ændre spillerens loss condition til at være at han har tabt.
Forudsætninger:
- En spiller er blevet oprettet

Testprocedure:
- Spillerens loss condition sættes til at spilleren har tabt.

Forventede resultater:
- Spilleren har ikke tabt når den bliver oprettet.
- Spilleren har tabt når loss conditionen ændres til at spilleren har tabt

Faktiske resultater:
- Som forventet har spilleren ikke tabt når spilleren bliver oprettet.
- Som forventet har spilleren tabt når spillerens loss condition bliver ændret til at spilleren har tabt.

Status: Bestået

Test Case: **testFieldmove**
Test Case ID: 39
Test Case beskrivelse: Formålet med denne test er, at teste at spilleren bliver rykket korrekt rundt på brættet.
Forudsætninger:
- En spiller er oprettet.

Testprocedure:
- Spilleren bliver rykket 6
- Spilleren bliver rykket 15
- Spilleren bliver rykket 12

Forventede resultater:
- Spillerens nuværende felt er lig 6 efter at have rykket 6.
- Spillerens nuværende felt er lig 21 efter at have rykket yderligere 15
- Spillerens nuværende felt er lig 12 efter at have rykket yderliger 12

- Spillerens forrige felt er lig 21.

Faktiske resultater:

- Som forventet er spillerens felt lig 6 efter at have rykket 6
- Som forventet er spillerens felt lig 21 efter at have rykket yderligere 15
- Som forventet er spillerens felt lig 12 efter at have rykket yderligere 12
- Som forventet er spillerens forrige felt lig 21.

Status: Bestået

Test af Account, foretaget i AccountTest.java

Test Case: **testDeposit**
Test Case ID: 40
Test Case beskrivelse: Formålet med denne test er, at teste at deposit metoden i account klassen virker efter formålet.
Forudsætninger:

- Der er oprettet en account.

Testprocedure:

- Der forsøges at indsætte et negativt beløb.
- Der forsøges at indsætte et positivt beløb.
- Der forsøges at indsætte et negativt beløb.

Forventede resultater:

- Balancen har ikke ændret sig
- Balancen er blevet forøget med det indsatte positive beløb
- Balancen har ikke ændret sig.

Faktiske resultater:

- Som forventet er balancen uændret
- Som forventet er balancen blevet forøget med det indsatte positive beløb
- Som forventet er balancen uændret.

Status: Bestået

Test Case: **testWithdraw**
Test Case ID: 41
Test Case beskrivelse: Formålet med denne test er, at teste at withdraw metoden virker efter hensigten. Dette inkludere at den skal returnere true hvis balance forbliver over 0, samt returnere false hvis man prøver at hæve flere penge end man har.
Forudsætninger:

- Der er oprettet en account.

Testprocedure:

- Der trækkes et positivt beløb som er lavere end balancen
- Der trækkes et negativt beløb.
- Der trækkes 0 fra kontoen.

- Der trækkes et positivt beløb som er lig med balancen.
- Der trækkes et positivt beløb som er højere end balancen.

Forventede resultater:

- Balancen bliver reduceret med det positive beløb og metoden returnere true
- Balancen forbliver uændret og metoden returnere false
- Balancen forbliver uændret og metoden returnere true
- Balancen bliver reduceret til 0 og metoden returnere true
- Balance bliver sat til 0 og metoden returnere false.

Faktiske resultater:

- Som forventet er balancen reduceret og metoden returnere true
- Som forventet er balancen uændret og metoden returnere false
- Som forventet er balancen uændret og metoden returnere true
- Som forventet er balance reduceret til 0 og metoden returnere true
- Som forventet er balancen blevet sat til 0 og metoden returnere false.

Status: Bestået

Test Case: **testTransfer**

Test Case ID: 42

Test Case beskrivelse: Formålet med denne test er, at teste at transfer metoden virker som forventet. Herunder at den returnere true hvis man prøver at flytte et beløb der er mindre end hvad en spiller har stående på sin konto, samt at den returnere false hvis man prøvet at flytte et beløb der er større end det der er på en konto.

Forudsætninger:

- To kontoer er blevet oprettet

Testprocedure:

- Der flyttes et beløb der er mindre end hvad der står på konto1, fra konto1 til konto2
- Der flyttes et negativt beløb fra konto1 til konto2
- Der flyttes 0 penge fra konto1 til konto2.
- Der flyttes et beløb der er lig det der står på konto1, fra konto1 til konto2
- Der flyttes et beløb der er større end det der står på konto1, fra konto 1 til konto2.

Forventede resultater:

- Beløbet er blevet overført og metoden returnere true
- Ingen af de to kontoers balance er ændret, og metoden returnere false.
- Ingen af de to kontoers balance er ændret, og metoden returnere true.
- Beløbet er blevet overført og metoden returnere true
- Den mængde penge der stod på konto1 er blevet overført til konto2, og metoden returnere false.

Faktiske resultater:

- Som forventet er beløbet overført og metoden returnerede true
- Som forventet er balancerne uændret og metoden returnerede false

- Som forventet er balancerne uændrede og metoden returnerede true
- Som forventet er beløb blevet overført og metoden returnerede true
- Som forventet er den mængde penge der stod på konto1 blevet overført til konto2 og metoden returnerede false.

Status: Bestået

Manuelle tests

Udover de automatiske tests der er beskrevet ovenfor, så har vi i løbet af hele udviklingen foretaget manuelle tests af hele spillet samt de enkelte metoder vi har implementeret. Dette er gjort ved at fjerne den del af koden der automatisk generer terningslag, og istedet indsætte nogle linjer kode der gør det muligt for brugeren af programmet at indtaste hvor langt en spiller skulle rykke sig. Disse manuelle tests er foretaget på alle dele af programmet, som man normalt vil komme til at benytte gennem et spil. En del af disse test cases kan ses nedenfor.

Test af Jail

Test case:
Test at processen ved at lande på et fængselsfelt virker som forventet.

Hvad forventes:
Der forventes at spilleren rykkes over på 'fængsels - på besøg' feltet som det eneste der sker denne tur. Når det bliver spillerens tur igen, kan spilleren vælge at købe sig ud af fængselsfeltet for 1000,- eller slå 2 ens for at komme ud. Slår spilleren ikke 2 ens, forbliver spilleren i fængslet, hvorefter spilleren løslades hvis spilleren har siddet i fængslet i 3 ture og der betales 1000,-. Disse kombinationer af tilfælde testes

Fremgangsmåde:
- En spiller lander på et fængselsfelt.
- Samme spiller køber sig ud når det bliver spilleren tur igen
- En ny spiller lander på fængsletsfeltet og slår 2 ens inden spilleren bliver smidt ud.
- En ny spiller prøver at slå sig ud 1 gang, og betaler sig derefter ud.
- En ny spiller prøver at slå sig ud 2 gange, og betaler sig derefter ud.

Resultat:
Alt virker som det skal.

Test af Chancecard

Test case:
Test at processen ved at lande på et 'Prøv lykken' virker som forventet.

Hvad forventes:

Der forventes at et en tilfældig proces mellem 30 forskellige processer udføres. Der er 6 antal forskellige typer af processer:

- Ryk bestemt antal felter.
- Betal et bestemt antal penge.
- Modtag et bestemt antal penge.
- Spilleren sendes i fængsel.
- Ryk til bestemt felt.
- Ryk til nærmeste rederi.

Fremgangsmåde:

- En spiller lander på et Chance-felt.

Resultat:
Alle Chancekort virker som forventet.

Test af Brewery

Test case:
Test at processen ved at lande på et bryggerifelt virker som forventet.

Hvad forventes:
Der findes 3 forskellige scenarier for at lande på et bryggerifelt:

- Feltet er ikke ejet, og det kan købes. Vælger spilleren ikke at købe det auktioneres det til de andre spillere.
- Feltet er ikke ejet og spilleren køber feltet.
- En spiller lander på et ejet felt, og spilleren slår igen med terninger. Der betales derefter 100*ejet antal bryggerier * terningeøjne til ejeren fra spilleren.

Fremgangsmåde:

- En spiller lander på et bryggerifelt, men vælger ikke at købe det
- En spiller lander på et bryggerifelt, og køber feltet
- En spiller lander på et felt som allerede er ejet, hvor ejeren ejer 1 bryggeri
- En spiller lander på et felt som allerede er ejet, hvor ejeren ejer 2 bryggerier

Resultat:
Auktion er en test i sig selv og er vist nedenunder. Derfor beskæftiger denne test case sig kun om hvorvidt de rigtige penge bliver trukket når en spiller landet på et ejet felt eller et ikke ejet felt, og begge cases virker som de skal.

Test af Shipping

Test case:
Test af processen ved at lande på et rederi virker som forventet.

Hvad forventes:

Der findes 3 forskellige scenarier for at lande på et bryggerifelt:

- Feltet er ikke ejet, og det kan købes. Vælger spilleren ikke at købe det auktioneres det til de andre spillere.
- Feltet er ikke ejet og spilleren køber feltet.
- En spiller lander på et ejet felt, og spilleren slår igen med terninger. Der betales derefter penge til ejeren fra spilleren alt efter hvor mange andre rederier ejeren ejer. Der betales:

● 500 for 1 rederi
● 1000 for 2 rederier
● 2000 for 3 rederier
● 4000 for 4 rederier

Fremgangsmåde:

- En spiller lander på et rederifelt, men vælger ikke at købe det
- En spiller lander på et rederifelt, og køber feltet
- En spiller lander på et felt som allerede er ejet, hvor ejeren ejer 1 bryggeri
- En spiller lander på et felt som allerede er ejet, hvor ejeren ejer 2 bryggerier
- En spiller lander på et felt som allerede er ejet, hvor ejeren ejer 3 bryggerier
- En spiller lander på et felt som allerede er ejet, hvor ejeren ejer 4 bryggerier

Resultat:
Auktion er en test i sig selv og er vist nedenunder. Derfor beskæftiger denne test case sig kun om hvorvidt de rigtige penge bliver trukket når en spiller er landet på et ejet felt eller et ikke ejet felt, og begge cases virker som de skal.

Test af Start

Test case:
Test af at Startfeltet virker som den skal.

Hvad forventes:
En spiller der passere start skal modtage 4000,-. Passeres der dog via fængsel skal der ikke modtages penge.

Fremgangsmåde:

- En spiller passere start normalt. Her skal der modtages penge
- En spiller passere start via en chancefelt. Her skal der modtages penge.
- En spiller passere start ved at blive sendt i fængsel og rykkes til ‘fængsel - besøg’. Her skal der ikke modtages penge.

Resultat: Det virker formidabelt! Dog kan man ikke rykke tilbage over start, hvilket heller ikke er et issue da vores prøvlykken kort kun rykker spilleren tilbage 2 felter og ikke 3, derfor kan man på intet tidspunkt komme til at rykke tilbage over start.

Test af Tax

Test case:
Test af at Skatfeltet virker som den skal.

Hvad forventes:
En spiller der lander på dette felt skal vælge at betale enten 10% af alt hvad man ejer eller 4000,- hvis der landes på det ene felt eller blot 2000 ,- for det andet.

Fremgangsmåde:

- En spiller lander på feltet og vælger at betale 4000,-
- En spiller lander på feltet og vælger at betale 10% hele personens værdi
- En spiller lander på det andet Skat felt og skal betale 2000,-

Resultat:
Skat feltet virker som det skal.

Test af Parking

Test case:
Test af at Parkeringsfeltet virker som den skal.

Hvad forventes:
En spiller der lander på dette felt skal modtage 4000,-

Fremgangsmåde:

- En spiller lander på feltet

Resultat:
Parkeringsfeltet virker som den skal.

Tests af Auction

Test case: Test af processen ved en auktion som virker som forventet

Test at auktion gennemfører som forventet.

Testen er gennemført idet programmet var i test fasen.

Hvad forventes:

At alle spillere, udover den spiller der har sagt nej til at købe feltet fra starten og spillere der er udgået, får lov til at byde på feltet.
At idet en spiller ikke længere vil byde udgår fra de resterende auktioner.
At man ikke kan byde under det tidligere bud.
At man ikke kan byde over sin balance
At man kan byde det der minimum bliver oplyst og at man kan byde alle sine penge.
At den spiller der byder højest vinder auktionen og bliver tildelt feltet.
At den spiller der vinder buddet bliver fratrukket det bud der er afgivet fra sin balance.

Fremgangsmåde:
Testeren kan lande på et vilkårligt felt det er muligt at eje.

- Tryk på knappen kast
- Indtast et tal der gør at spilleren lander på et felt der kan ejes. Her kan anbefales 1.
- Afslå at købe feltet. Nu burde auktions systemet starte.

Resultat af test
Testen forløb som forventet på alle punkter udover et:

- Det er ikke muligt med det første bud at byde minimumsbudget. Dvs. hvis minimumsbudget er på 1200, er det ikke muligt at byde 1200, selvom det er forventet at man kan starte med at byde til udbudsprisen.

Konfiguration

Når et softwareprodukt rulles ud til kunden, kan der opstå mange kompabilitetsproblemer. Dette kan både være i forhold til styresystemer, krævede runtime environments og andre installerede programmer.

Som udviklingsplatform er der i projektet brugt Eclipse Luna med Java 7. Der er dog blevet udviklet på forskellige styresystemer som vi vil beskrive nærmere forneden.Desuden er der også brugt et grafisk brugerflade bibliotek kaldet GUI.jar.

Produktionsplatformen er i dette projekt det samme som udviklingsplatformen.

Krav til gennemkørsel af det udviklede spil, nedenfor nævnte er de kontrollerede konfigurationer:

- Windows 8.1, OS X v10.10 Yosemite eller Ubuntu 14.04.

- Programmet kan muligvis også køres på andre styresystemer men det kan ikke loves.

- Java 7
  - Programmet kan muligvis også køres på andre udgaver af Java, disse er dog ikke testet.

- Eclipse Luna
  - Andre udgaver af Eclipse kan muligvis også godt køre spillet, dette er ikke testet.

Importering af projekt i Eclipse og kørsel

For at åbne programmet downloades og installeres Eclipse Luna. Programmet åbnes i Eclipse og importeres som et projekt. I Eclipse trykker man på "File", derefter "Import". På den side der da åbner, vælges "General" og derefter dobbeltklikker man på "Existing Projects into Workspace".

På denne side trykker man på "Browse" øverst i højre hjørne, og derefter navigeres til mappen hvori man har gemt projektet. Når det er gjort, trykker man på OK og projektet burde da gerne vises i det store hvide felt. Hvis det gør det, kan projektet derefter importeres til Eclipse ved at trykke på "Finish", og projektet vises i Project Exploreren i venstre side af Eclipses Project Explorer, og spillet kan afspilles ved at navigere til Game-klassen og derefter trykke på knappen "Run".

Kompilering af projekt til eksekverbar .JAR fil

En .JAR-fil er en pakke med filer, i stil med en ZIP-fil, der indeholder kompileret kode. En .JAR fil er oftest et program hvor det er indkodet i filen hvilken klasse der skal køres når man eksekverer filen, altså hvilken klasse der har en main-metoden.

Export til eksekverbar .JAR fil

For at kompilere spillet til en eksekverbar .JAR fil, åbnes Eclipse Luna. Dernæst trykker man på File > Export. Her kan man så vælge hvilken type fil man vil eksportere til. I dette tilfælde ønskes Java > Runnable JAR file. Man vælger herefter i "Launch configuration" hvilken klasse der har main-metoden. I dette tilfælde vil det være "GameController - 11_final". Man vælger dernæst i "Export destination" hvad

.JAR filen skal hedde samt hvor den skal lægges på ens drev. Man trykker nu “Finish” og hvis der kommer en prompt op der spørger, om operationen skal ompakke nødvendige biblioteker, tryk “OK”.

Der er i projektet vedlagt en eksekverbar .JAR fil som resource som er testet på DTU’s maskiner i databaren.

Konklusion

Vi har i løbet af projektet fået udviklet et stort set fuldt ud funktionelt matadorspil. Vi har arbejdet sammen i en gruppe og gruppearbejdet har fungeret rigtig godt. Vi har forbedret vores evner til at kode større JAVA projekter, og vi er blevet langt bedre til at arbejde projektorienteret i gruppen.

Der er enkelte funktioner i spillet vi ikke har udviklet men dette er stort set ubetydelige funktioner for spillets overordnede afvikling. Vi har fået compiled en eksekverbar .JAR fil som kører på DTU's maskiner i databaren, hvilket var et af kravene.

Vi mener selv, at vi har opfyldt alle læringsmålene for kurset i løbet af vores projekt, og kan altså konludere at projektet har været vellykket og læringsrigt.