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

---

CDIO del 2 - Gruppe 42

Christoffer Perch Nielsen
s202125

Emil Nymark Trangeled
s195478

Stig Bødtker Petersen
s186333

Mohammed Zahed
s186517

Mads Emil Kaas Hansen
s195159

Magnus Thorup Müller Larsen
s194056

Afleveringsfrist 30. oktober 2020

Link til github repository der blev brugt under projektet:
https://github.com/DTU1semHackerChamps/42_del2

Timeregnskab

Christoffer Perch Nielsen : 22 timer

Emil Nymark Trangeled (En del af commits er lavet sammen med Magnus over teamviewer): 29 timer

Stig Bødtker Petersen : 25.5 timer

Mohammed Zahed : 25 timer

Mads Emil Kaas Hansen : 25 timer

Magnus Thorup Müller : 25 timer

Indholdsfortegnelse

Resume

Vores CDIO del 2 projekt, består af et brætspil mellem to spillere. Spillerne skiftes med at tage tur, hvor de skal hver især kaste med to terninger. Værdien af de to terninger bliver lagt sammen og spilleren rykker frem et af de 12 felter der er tilsvarende summen af terningerne. Felterne har en prædefineret effekt som kan påvirke spillernes pengebeholdning. Spillerne starter med en pengebeholdning på 1000. Og den første der når en pengebeholdning på 3000 vinder spillet.

Indledning

Dette projekts formål er at skabe et brætspil efter kundes vision. Spillet skal være simpelt og let at spille uden nogen form for introduktion på forhånd[1]. Inden koden til spillet bliver skrevet opstiller vi vores use cases og udarbejder de nødvendige modeller og diagrammer som f.eks. klasse, domæne, sekvens-diagrammer osv for at udarbejde den rette fremgangsmåde til udvikling af koden. Efter implementeringen af koden bliver diagrammerne revurderet for at skabe en bedre sammenhæng mellem kode og udviklings modellerne. Til sidst foretager vi nogle tests af programmet ved hjælp af Junit 5, for at sikre, at vi leverer et ordentligt produkt.

Kravspecifikation

- Der skal være to spillere
- Slå med to terninger på en gang
- Spilleplade med felter som har numrene 2-12.
- Summen af terningerne afgøre hvilket felt man lander på
- Hvert felt kan give point eller gøre så man mister point, afhængig af hvor man lander.
- Spilleren skal indeholde en score
- Score pr. spiller skal vises
- Spillerne starter med 1000 point.
- Vinderen er fundet når den første person når 3000 point.
- En spillers score må aldrig komme under 0
- Der skal være en mulighed for genstart af spillet efter en spiller har vundet spillet
- Der udskrives en tekst når man lander på et felt
- Spillet skal kunne spilles på DTU's databarer
- Felterne skal have de navne og effekt som er givet i feltlisten.
- Spillerne skal beholde deres position efter de har landet på et felt
- Kunden skal bruge jdk 14 +
- Kunden skal bruge windows 10 +
- Kunden skal bruge en java compiler

[1] CDIO 1 - https://docs.google.com/document/d/1-XGd8bKliHMJeK0lLX8EHQ7FTbhvrboE7ldwqu_zQbQ/edit

Feltliste

1.	(Man kan ikke slå 1 med to terninger)		
2.	Tower	+250	
3.	Crater	-100	
4.	Palace gates	+100	
5.	Cold Desert	-20	
6.	Walled city	+180	
7.	Monastery	0	
8.	Black cave	-70	
9.	Huts in the mountain	+60	
10.	The Werewall (werewolf-wall)	-80,	men spilleren får en ekstra tur.
11.	The pit	-50	
12.	Goldmine	+650	

Use cases[2]

- Spil spillet
- Genstart spil

Use case: SpilSpillet
ID: 1
Brief description: Man trykker spil, derefter bliver der oprettet en spiller_1 og spiller_2.
Primary actors: Spiller 1 Spiller 2
Secondary actors: Ingen.
Preconditions: Ingen.
Main flow: 1. Man trykker spil på startsiden. 2. Derefter genererer spillet 2 spillere, så man kan skiftes om at slå med terningerne. 3. Spillerne lander på et felt på spil brædet baseret på hvad de slog med terningerne. 4. spillerne får en effekt af point eller en ekstra tur baseret på hvilket felt de lander på. 5. spillet fortsætter indtil der er en spiller der når en points værdi på 3000
Postconditions: spillet erklære en vinder når en spiller når 3000 point
Alternative flows: Ingen.

[2] https://docs.google.com/document/d/1-XGd8bKliHMJeK0lLX8EHQ7FTbhvrboE7ldwqu_zQbQ/edit

Use case: GenstartSpil
ID: 2
Brief description: Spillerne genstarter spillet, når de har lyst til.
Primary actors: Spiller 1 Spiller 2
Secondary actors: Ingen.
Preconditions: 1. Spillerne skal have startet spillet før de kan genstarte spillet. 2. Der skal være erklæret en vinder ved 3000 points
Main flow: 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

Analyse

Risikoanalyse[3]

- 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, ellsers 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-XGd8bKliHMJeK0lLX8EHQ7FTbhvrboE7ldwqu_zQbQ/edit

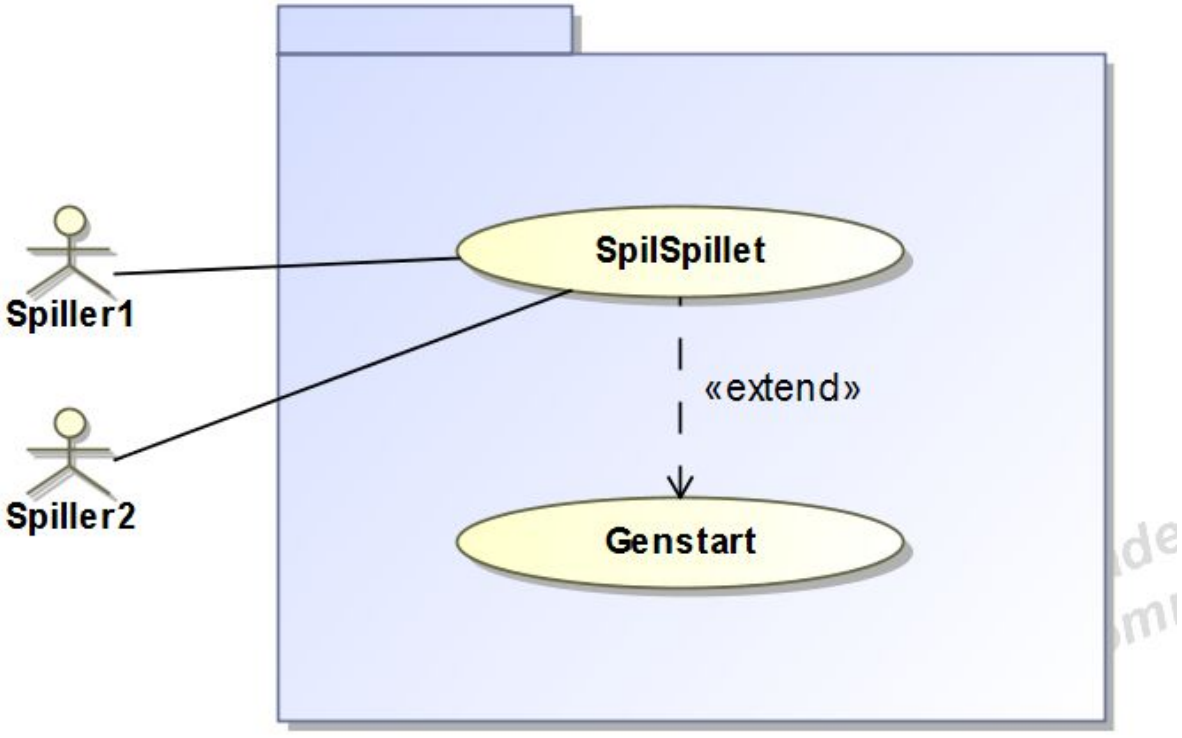
Klasser

Vi har defineret følgende klasser:

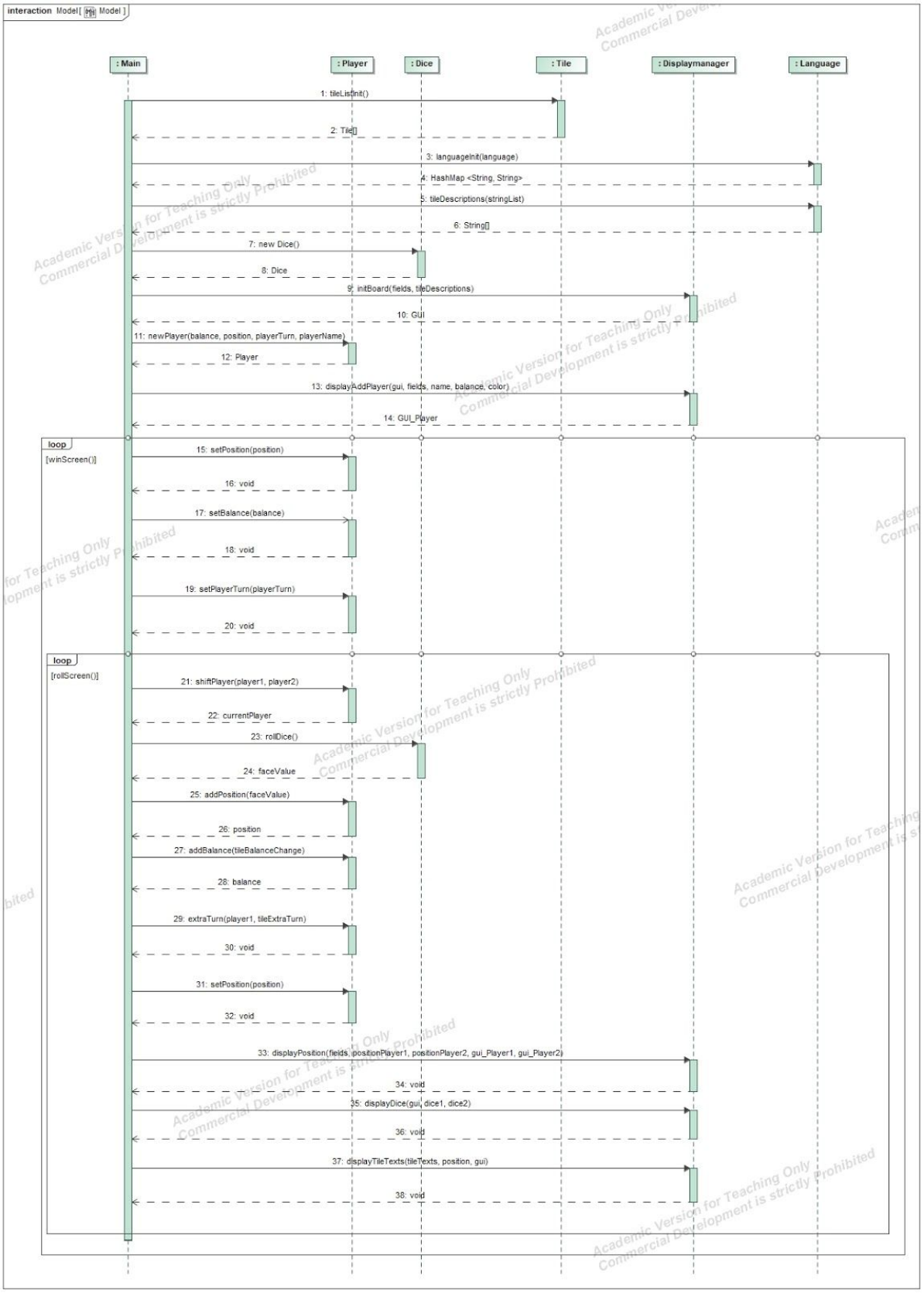
- Main
- Dice
- Player
- Displaymanager
- Tile
- Language

Design

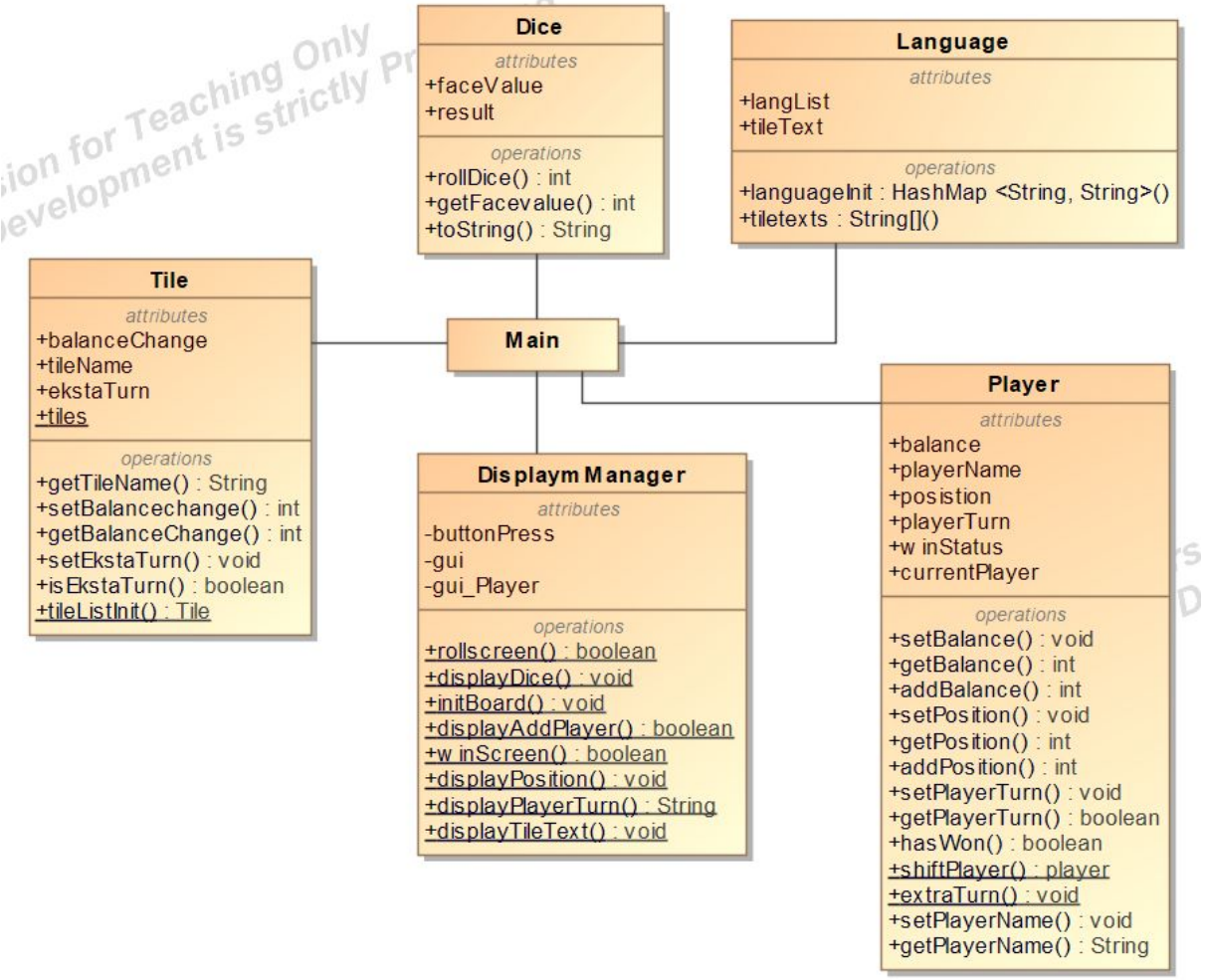
Use case diagram:



System Sekvensdiagram:



Design klassediagram:



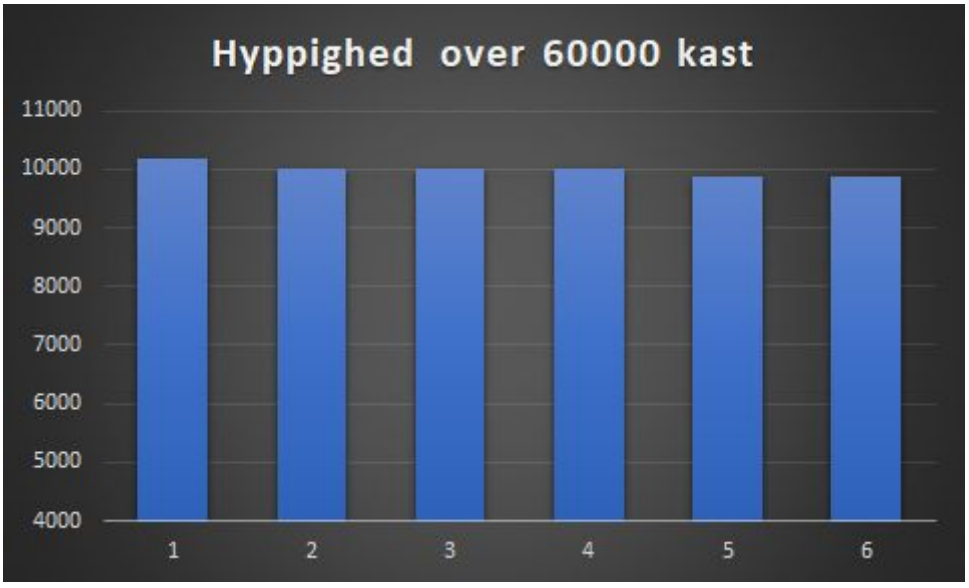
Implementering

Til at versionsstyre projektet har vi brugt Git. Vi har valgt at have en branch struktur med main, developer og feature-branches, hvor de er baseret på developer. Vi har derfor valgt at lave en ny branch, for hver feature vi starter på. Dog skal det siges, at vi har valgt at arbejde på en feature-branches der er defineret ud fra de klasser vi fandt der skulle bruges til spillet. Når en feature blev færdig, vælger vi at merge den pågældende feature ind i developer-branchen. Når developer-branchen, så er 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.[4]

[4] CDIO 1 - https://docs.google.com/document/d/1-XGd8bKliHMJeK0lLX8EHQ7FTbhvrboE7ldwqu_zQbQ/edit

Test

I denne test er der blevet slået med en 6 sidet 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 “passede”.



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

Projektplanlægning

Vi startet 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 klassediagram.

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.

Brugervejledning

1. Brugeren skal først downloade og installere IntelliJ IDE'en, hvilket kan gøres her: https://www.jetbrains.com/idea/download
2. Derefter skal brugeren downloade og installere en JDK. Her anbefaler vi OpenJDK 15. (man kan installere den fra IntelliJ)
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 copy paste linket til Github repositoriet https://github.com/DTU1semHackerChamps/42_del2 og derefter skal man trykke på "Clone".
4. Hvis Junit ikke er importeret, skal man ind under test klasserne og importerne Junit.
5. Hvis GUI'en ikke virker, skal man vælge "matadorgui-3.1.6.jar" og tilføje den til ens library.
6. Nu skal brugeren gå ned og højreklikke på main, trykke run program og køre det. Programmet starter derefter og brugeren kan begynde at spille.

Konklusion

Vi kan konkludere, at vi har formået at lave et brætspil program, der opfylder alle de krav der blev stillet i kravspecifikationen. Gruppen har igen genbrugt den GUI der blev udleveret i undervisningen, men har modificeret den. Ud fra klasse, domæne, sekvensdiagrammerne der er blevet lavet, har gruppen haft et solidt fundament til at skabe programmet.
Programmet er simpelt og ligetil og der behøves ikke nogen introduktion for at spille det, og med det kan der konkluderes at vi har lavet et produkt der virker.

Bilag

Litteraturliste

CDIO 1
https://docs.google.com/document/d/1-XGd8bKliHMJeK0lLX8EHQ7FTbhvrboE7ldwqu_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