

Eksamensprojekt Cards against humanity

Simon S. Erfurth 2.B
Odense Tekniske Gymnasium

2. maj 2014



Figur 1: Forsiden når man åbner programmet.

Indhold

1	Formål & projektbeskrivelse	3
2	Afgrænsning	3
3	Analyse & Design	3
3.1	Klassediagrammer	3
3.2	Overvejelser omkring kort	5
4	Udvikling	6
4.1	Udseende af MainClass	6
4.2	Brug af cardlayout	7
4.3	Metoderne Player og Cards	8
4.4	Metoden Creatdeck	9
4.5	Alternativer til metoden <code>opdaterSide2()</code>	10
4.6	Variabel liste	11
5	Præsentation	11
6	Diskussion	13
7	Konklusion	14
8	Kilder	15
9	Billag	15
9.1	Bruger historier	15
9.1.1	Brugerhistorie 1 - Antal spillere	15
9.1.2	Brugerhistorie 2 – Oprettelse af spillerne	16
9.1.3	Brugerhistorie 3 – Kortene	16
9.1.4	Brugerhistorie 4 – En runde	17
9.1.5	Brugerhistorie 5 – Afslutningen af et spil	17
9.1.6	Brugerhistorie 6 – Spil igen	18
9.2	Koden	18
9.2.1	Den primære klasse - MainClass	18
9.2.2	Data klassen Cards	30
9.2.3	Data klassen Player	32

1 Formål & projektbeskrivelse

Formålet med projektet er at skrive spillet kendt som "Cards Against Humanity" i java. Programmet skal køre på en computer/tablet, og understøtte 4 til 8 spillere.

Som udgangspunkt er det kun "basic rules" der skal implementeres (ekls. "Pick 2" rules), men i fremtiden ville det også være en mulighed at implementere "house rules" eller "Pick 2".¹

Helt kort er de regler som spillet skal følge som følgende. Hver spiller trækker 10 tilfældige kort. Der vil så vær en spiller der starter med at skulle læse et tilfældigt valgt spørgsmålskort op. De andre spillere skal så på tur vælge et af deres 10 kort. Når alle har valgt deres kort skal den spiller der læste det sorte kort op så vælge hvilket af de hvide han synes var sjovest. Den spiller, hvis kort han vælger får så et point. En ny spiller skal nu læse et spørgsmåls kort op. Spillet går på runde indtil de enten stopper spillet eller der ikke er flere kort. Man vil så kunne se hvem der har fået flest point.

2 Afgrænsning

Det er urealistisk at vi kan nu at lave hele spillet i den tid der er til rådighed. Derfor vil der i dette projekt kun blive lavet valget af antal spillere, navngivning af spillere og oprettelse af spillere samt de 2 dæk med sorte og hvide kort som man spiller med.

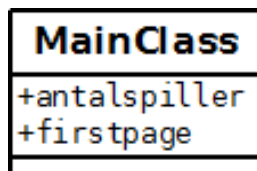
3 Analyse & Design

3.1 Klassediagrammer

Efter at have skrevet brugerhistorie ¹² gik vi i gang med at tegne et klasse diagram til denne. Indtil videre var der ingen grund til at lave flere klasser end 1, se figur 2.

¹For komplette regler se http://s3.amazonaws.com/cah/CAH_MainGame.pdf

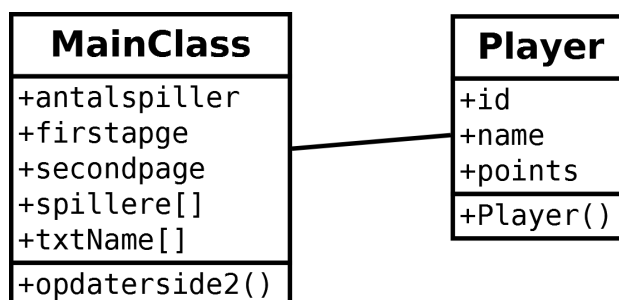
²Alle bruger historier kan ses i billagene, og vil her blot blive refferet til som „brugerhistorie x“.



Figur 2: Klassediagrammet til vores første iteration.

Indtil viddere er der ikke meget vi skal tage højde for, og vi kunne meget hurtigt have lavet et program som kunne det vi havde brug for. Dog valgte vi at lave programmet som et card layout, så vi nemt kunne skifte til en ny side senere.

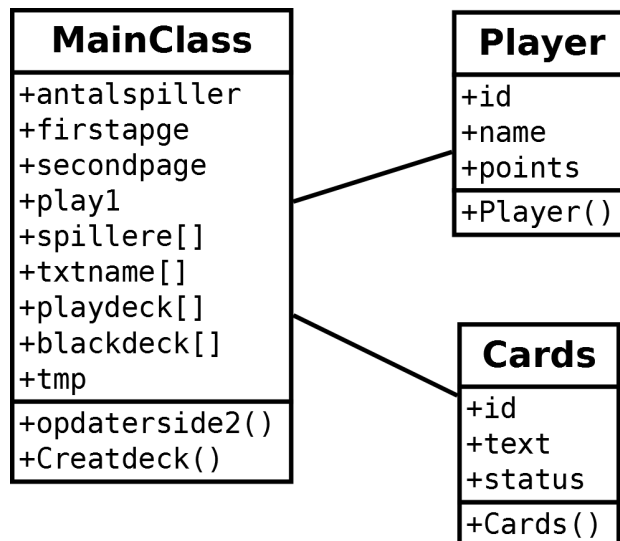
Da vi senere havde implementeret og testet dette gik vi i gang med at lave et klassediagram som passede til brugerhistorie 2, her var vores mål at vi skulle have oprettet spillerne, og da de havde en del egenskaber tilfælles valgte vi at lave en klasse til dette, se figur 3.



Figur 3: Klassediagrammet under 2. iteration.

Netop at vi lavede en klasse specifik til spillere gav os den fordel at alle variabler (eller i dette tilfælde lister) man opretter af denne type får automatisk egenskaberne som denne klasse har. I dette tilfælde egenskaberne `id`, `name` og `points`. Dette er som sagt en fordel når vi har flere spillere der alle skal have disse egenskaber.

Tilsvarende udvided vi vores klassediagram da vi skulle til at lave 3. iteration for den tilsvarende brugerhistorie. Her var målet at få lavet de to dæk der skulle være i spillet, og vi valgte derfor at lave en ny klasse til kort, dette blev til klassen `Cards`. Vi tilføjede igen en række variabler og lister til vores `MainClass`, for at have plads til de ting vi implementerede, se figur 4.



Figur 4: Det endelige klassediagram som vi fik anvendt, tilhøre itteration 3.

Her har vi lavet metoden `Cards()` i klassen af samme navn. Dette gør at vi får en metode hvor vi skal angive en række ting hver gang vi skal oprette et nyt kort, hvilket er en fordel i det det betyder vi helt sikkert får alt angivet med en standard value hver gang vi oprette et nyt kort. Se mere om dette i afsnittet om implementering.

3.2 Overvejelser omkring kort

For at få lavet et stort dæk med mange kort er det nødvendigt at vi har et sted at hente teksten fra. Jeg havde haft flere ideer om hvor man kunne opbevare dataen. En ide var at lave en lang metode som oprettede alle kortene, en for en, med en ny string og et nyt nummer hver gang. En anden var at have et excel dokument, hvor at hver box i en linje var de hvide kort og hver box i en anden linje var de sorte kort³.

I sidte ende var vi kommet frem til at det nemmeste var at have gemt alle kort tekstne i 2 forskellige dokumenter, et til de hvide og et til de sorte. Vi kunne så importere dem individuelt⁴, og oprette lister til de to dæk, som

³Inspireret af <http://mrbool.com/reading-excel-file-with-java/24562>

⁴Inspirert af:
<http://stackoverflow.com/questions/3853628/>

havde den linjes tekst der svare til deres `id + 1`. Vi valgte denne metode da den virkede noget simplere end excel metoden og gav en langt større frihed for at skifte filer til andre dæk, altså spille med andre kort, end den første metode gav.

4 Udvikling

4.1 Udseende af MainClass

Efter vi har oprettet vores MainClass vælger vi at give den layoutet „Card-Layout“

```
1 frame.getContentPane().setLayout(new CardLayout(0, 0));
```

Vi vælger at give vinduet et cardlayout, da det gør at det er nemt at ændre på hvad der er i forgrunden af det vindue vi arbejder i. De to tal vi angiver i paranteserne til sidst er hvor mange pixel vi vil have mellem vores cardlayout og kanten på vores vindue. Den første er verticalt, og der vil være det gab mellem toppen af vinduet og toppen af cardlayoutet, og mellem bunden af dem. Det andet tal er mellem højre og venstre kanterne på vinduet og cardlayoutet. Vi har sat begge disse til 0 da vi vil have vores cardlayout til at fylde hele vinduet.

```
1 frame.setBounds(100, 100, 900, 600);
2 frame.setTitle("Cards Against Humanity");
```

Vi vælger med metoden `frame.setTitle()` hvad det der skal stå i toppen af vores vindue skal stå, oftest titlen på programmet man bruger og evt. filen man arbejder med. I dette tilfælde har vi naturligvis valgt at skrive "Cards Against Humanity", da dette er navnet på spillet, og det vil gøre vinduet let at identificere i fx. joblisten hvis det crasher.

Metoden `frame.setBounds()` angiver de bånd som vores vindue har når man åbner det. I dette tilfælde vælger vi at det skal starte i et punkt der er placeret i (100,100) dette gør vi for at undgå problemer hvis man har sin proceslinje i venstre side af skærmen (som jeg har), eller øverst i skærmen.

Vi vælger så at den skal have en brede af 900 pixel og en højde af 600 pixel. Højden er valgt da de mindste skærme som folk stadigvæk bruger (normalt) har en højde af 768 pixel, og da vi starter 100 pixel nede og så går yderligere 600 ned er der stadigvæk 68 til overs til proceslinje.

```
1 JLabel lblWelcome = new JLabel("CARDS AGAINST HUMANITY");
2 lblWelcome.setBounds(115, 5, 654, 63);
3 lblWelcome.setForeground(Color.DARK_GRAY);
4 lblWelcome.setFont(new Font("Tahoma", Font.PLAIN, 52));
5 firstpage.add(lblWelcome);
```

Her opretter vi overskriften som er noget af det første man ser når man starter spillet. Den siger meget simpelt "CARDS AGAINST HUMANITY". Vi angiver herefter hvor den skal befinde sig og hvor stor den skal være med `.setBounds(x,y,w,h)` Først angiver vi hvor den skal starte med en x-koordinat, derefter med en y-koordinat. Vi angiver heretter hvor bred og høj den skal være med `width(w)` og `height(h)`.

For at gøre det tydeligt at det er overskriften sætter vi skrifttypen til at være en uden fødder og skriftstørrelsen til at være 52. Til sidst tilføjer vi den til vores første panel, `firstpage`.

4.2 Brug af cardlayout

At vi har valgt at bruge `CardLayout` gør at vi nemt kan skifte hvad der er i forgrunden. Dette anvender vi når vi trykker på næste knappen og programmet ændre på hvad brugeren ser.

Vi kan nu oprette en række panerler (cards) som vi kan skifte i mellem.

```
1 JPanel firstpage = new JPanel();
2 frame.getContentPane().add(firstpage, "Start");
3 firstpage.setLayout(null);
```

Når vi opretter et nyt panel starter vi med at definere at det er af typen `JPanel` og at vi giver det så et navn. Derefter tilføjer vi det til vores frame, så vi kan få det frem i dette vindue. Til sidst vælger vi at sætte layoutet til

`null`. Dette er det vi kalder absoul layout. Alternativt kunne vi have valgt grid layout eller have lavet et under cardlayout.

Når vi senere ønsker at skifte hvilke panel vi har forest i programmet på et senere tidspunkt gør vi dette med følgende to linjer kode:

```
1 c = (CardLayout)frame.getContentPane().getLayout();
2 c.show(frame.getContentPane(), "Navngivning");
```

Her bruger vi variablen `c` som en placeholder til at skifte koden. Vi starter med at definere at `c` er det nuværende layout vores vindue har, og vi ændre detn så til at være det nye layout vi ønsker skal være forest. I tilfældet ovenfor er det `Navngivning` panelet som er nummer 2 panel man ser. Vi gør dette inde i en knap, så det ændre sig når man trykker på den, så hvis vi havde en knap der hed `btnNewButton`, og som kun skulle skifte hvad man så ville koden til den se sådan her ud:

```
1 btnNewButton.addActionListener(new ActionListener() {
2     public void actionPerformed(ActionEvent arg0) {
3         c = (CardLayout)frame.getContentPane().getLayout();
4         c.show(frame.getContentPane(), "Navngivning");
5     }
6 }
7 );
```

4.3 Metoderne Player og Cards

Player og Cards har det til fælles at de begge er en klasse som også indeholder en metode af samme navn. Metoden tjener i begge tilfælde det formål at man kan oprette en variabel af typen `Player` eller `Cards`. Lad os kikke på metoden for `Player`.

```
1 public Player(int number, JTextField navn){
2     String navnet = navn.getText();
3     this.id = number;
4     this.name = navnet;
```



```
5         this.points = 0;  
6     }
```

Her starter vi med at angive at dette er en `public` metode, hvilket betyder at vi kan referere til den fra andre klasser, som fx. `MainClass`. Vi giver den så et navn, i dette tilfælde `Player`, og så angiver vi det som vi vil have oplyst med den, i dette tilfælde en `int` som vi kalder `number` og et `TextField` som vi kalder `navn`. Dette gør at når man anvender metoden skal man angive disse 2 egenskaber, so mvi så bruger når vi oprette metoden.

I dette tilfælde vælger vi at dennes `id` skal være lig nummeret, og at dennes `name` skal være det `TextField` de opgiver. i starter med at definere en string som det der står i navnet og vi sætter så `this.name` til at være dette navn.

Vi har en tilsvarende metode i klassen `Cards` som vi bruger når vi orpetter dækkene.

4.4 Metoden Creatdeck

Når man på side 2 har skrevet sine navne ind og trykker videre anvender programmet metoden `Creatdeck()`.

```
1 Creatdeck();
```

Dette aktivere metoden længere nede i programmet. Denne metode angiver størrelsen af de to dæk vi spiller med og tilføjer kort til dem. Den blander dem også.

```
1 playdeck = new Cards[10];  
2 for(int ii =0; ii <10;ii++){  
3     playdeck[ii]= new Cards(ii, "Svar nummer " + ii + ".");  
4 }
```

I dette tilfælde har vi bare valgt at lave dækket med en størrelse på 10, i den endelige udgave ville det være en fordel af lave det af den størrelse som svare til det antal linjer der er i den fil der indeholder teksten til kortene.

Vi anvender så en `for` sætning til at oprette alle kortne i lsiten. I den endelige version ville teksten naturligvis ikke skulle være „Svar nummer ii.“, men det som der skulle stå på kortet. Sandsynligvis ville selve `for` delen se sådan her ud:

```
1 for(int ii =0; ii <playdeck-length + 1;ii++)
```

Metoden `Cards` ligger i klassen af samme navn og den laver angver at kortets `id` bliver det nummer der er angivet først og at teksten på det bliver den string som er angivet som del nummer 2. Koden til dette ser sådan ud:

```
1 public Cards(int number, String data){
2     this.id = number;
3     this.text = data;
4 }
```

Vi slutter så af med at blande dækkene med et stykke kode som vi har fået stillet til rådighed fra <http://programmeringc.wikispaces.com/Sm%C3%A5+0pgaver>.

4.5 Alternativer til metoden `opdaterSide2()`

Som det er nu indeholder metoden `opdaterSide2()` en række `if` sætninger som kun bliver aktiveret hvis spillerne har valgt at de er flere end `x` spillere. et eksempel:

```
1 if(antalspillere>5){
2     txtName[6].setText("Name");
3     txtName[6].setColumns(10);
4     txtName[6].setBounds(320, 315, 86, 20);
5     secondpage.add(txtName[6]);
6
7     JLabel lblPlayer6 = new JLabel("Player 6:");
8     lblPlayer6.setBounds(246, 321, 66, 14);
9     secondpage.add(lblPlayer6);
10 }
```

Det eventsående stykke kode bliver kun aktiveret hvis der er flere spillere end 5, altså hvis spillerne vælger at de er 6, 7 eller 8.

Dette kunne være lavet på mindre plads og mere elegant, men en for sætning men pga. tidsmangel blev dette nedprioriteret.

4.6 Variabel liste

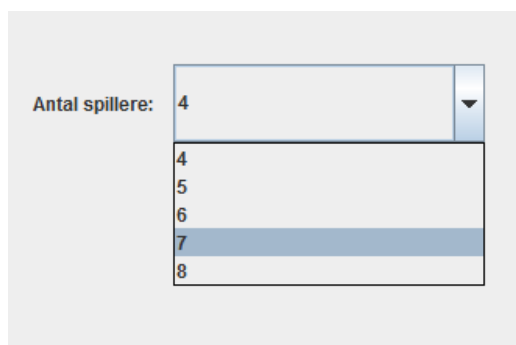
Her vil vi angive en liste af de relevante variable, deres type og en kort beskrivelse af hvad de gør.

Navn	Type	Beskrivelse
antalSpillere	int	Holder styr op hvor mange spillere der er i spillet.
antalSpiller	JComboBox	Combobox til at vælge antal spillere.
frame	JPanel	Vinduet til spillet.
firstpage	JPanel	Den første side, hvor man vælger antal spillere.
secondpage	JPanel	Den anden side, hvor man navngiver spillerne.
play1	JPanel	Den første side man skulle se når man spillede.
spillere[]	Player	Liste af spillere i spillet.
txtname[]	JTextField	Liste af navne på spillerne.
playdeck[]	Cards	Liste af de hvide kort i spillet.
blackdeck[]	Cards	Liste af de sorte kort i spillet.
tmp	Cards	En midlertidig variabel til at blande dækkene.
c	CardLayout	En placeholder til at skifte layout.

5 Præsentation

Programmet mangler meget for at være et færdigt produkt, men man kan dog starte spillet, vælge hvor mange spillere man er og navngive dem. I programmet bliver der også oprettet lister til dækkene, og oprettet nogle test kort i dem. Der blev aldrig importeret selve spille delen, ej heller point skærmen som der til sidst skulle angive hvem der vandt. Grunden til at vi ikke er nået længere skyldes primært at det var et meget omfattende projekt at skrive hele spillet, og dels at der var problemer med at finde den optimale måde at få teksten til kortene ind i spillet.

Når man starter spillet op bliver man mødt af teksten „CARDS AGAISNT HUMANITY“ og drop down menue, hvor man kan vælge hvor mange spillere man er, mellem 4 og 8 (Se figur 1 (forsiden) og figur 5).



Figur 5: Når man starter spillet skal man vælge hvor mange spillere man er.

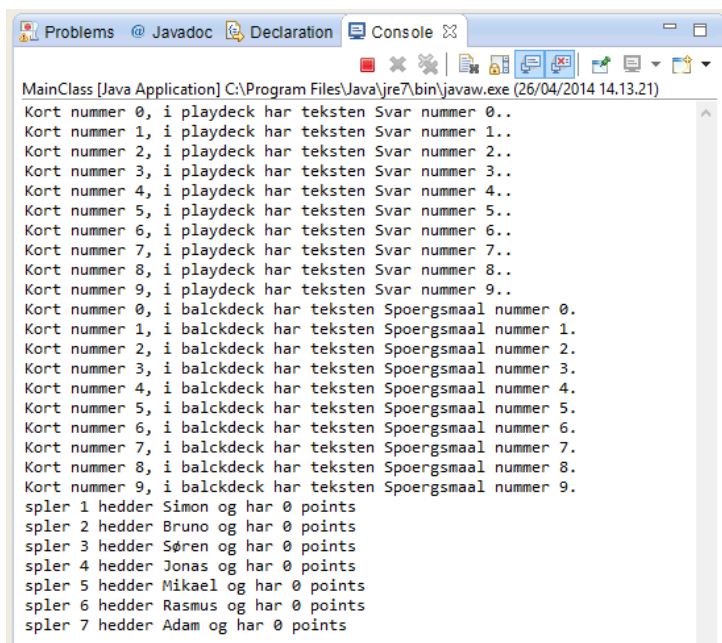
Alt efter hvor mange spillere man er vil side 2, som man kommer til når man trykker på næste blive opdateret til at indeholde det korrekte antal felter, hvor hver spiller så skal skrive det navn de ønsker at have. Alternativt kan de lade den stå blank og deres navn vil bare være „name“, men de har stadigvæk et individuelt spiller nummer, se figur 6.



Figur 6: Når man laver et nyt spil skal man navngive de spillere der deltager i spillet.

Når man trykke viddere her er der ikke mere synligt i spillet, men det

opretter 2 dæk, et til de hvide kort og et til de sorte kort, samt spillerne i en lsite. For at teste at disse blev oprettet korrekt fik vi programmet til at udskrive spillerne og kort i konsolen når den havde oprettet dem, se figur 7.



```
MainClass [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (26/04/2014 14.13.21)
Kort nummer 0, i playdeck har teksten Svar nummer 0..
Kort nummer 1, i playdeck har teksten Svar nummer 1..
Kort nummer 2, i playdeck har teksten Svar nummer 2..
Kort nummer 3, i playdeck har teksten Svar nummer 3..
Kort nummer 4, i playdeck har teksten Svar nummer 4..
Kort nummer 5, i playdeck har teksten Svar nummer 5..
Kort nummer 6, i playdeck har teksten Svar nummer 6..
Kort nummer 7, i playdeck har teksten Svar nummer 7..
Kort nummer 8, i playdeck har teksten Svar nummer 8..
Kort nummer 9, i playdeck har teksten Svar nummer 9..
Kort nummer 0, i balckdeck har teksten Spoergsmaal nummer 0.
Kort nummer 1, i balckdeck har teksten Spoergsmaal nummer 1.
Kort nummer 2, i balckdeck har teksten Spoergsmaal nummer 2.
Kort nummer 3, i balckdeck har teksten Spoergsmaal nummer 3.
Kort nummer 4, i balckdeck har teksten Spoergsmaal nummer 4.
Kort nummer 5, i balckdeck har teksten Spoergsmaal nummer 5.
Kort nummer 6, i balckdeck har teksten Spoergsmaal nummer 6.
Kort nummer 7, i balckdeck har teksten Spoergsmaal nummer 7.
Kort nummer 8, i balckdeck har teksten Spoergsmaal nummer 8.
Kort nummer 9, i balckdeck har teksten Spoergsmaal nummer 9.
spler 1 hedder Simon og har 0 points
spler 2 hedder Bruno og har 0 points
spler 3 hedder Søren og har 0 points
spler 4 hedder Jonas og har 0 points
spler 5 hedder Mikael og har 0 points
spler 6 hedder Rasmus og har 0 points
spler 7 hedder Adam og har 0 points
```

Figur 7: Konsollen med oprettelsen af dækkene og spillerne.

6 Diskussion

De dele af projektet som blev udført elver i høj grad op til de forventninger vi havde til projektet til at starte med. Desværre havde vi en forventnign om at vi kunne nå væsentlig mere end det ente med at være tildfældet. Dette resulterede i at det mest væsentlige i spillet, nemlig selve kort spillet, aldrig kom til at eksistere. Havde der været bedre tid, var det hvis alt forsatte som nu sansynligvis endt ud med et velfungerednde og generelt udnerholdende spil.

Hvis der var en ting som jeg gerne ville gå tilbage og ændre, var det min implementation af opdateringen af side 2. Her kunne vi med fordel have lavet det om til en `for` sætning, dette ville ikke bare have gjort koden mere

overskuelig, men det ville også have gjort det nemmere at gå tilbage og lave om på placeringerne af boxene, så side 2 var belvet mere symmetrisk.

I løbet af projektet har jeg lært en række ting. Først og fremmest er jeg blevet klogere på hvor lang tid det rent faktisk tager at få lavet et projekt, fra start til slut. Jeg har også lært at det ikke kan betale sig at lave alle bruger historier for programmet til at starte med, men hellere lave en 2-3 stykker og så implementere dem, og så kan man altid lave flere.

Rent programmerigns mæssigt har jeg lært, men aldrig anvendt, hvordan man kan importere tekst dokumenter og excel regneark ind i java. Dette skulle have været anvendt til at lave dækkene, men så langt kom vi desværre aldrig.

7 Konklusion

Vi har fået lavet en start af spillet „Cards agaisnt humanity“ i java. Det udnerstøtter 4-8 spillere og er test på PC, men der burde ikke være noget i vejen for at det kunne køre på en tablet.

8 Kilder

- <http://programmeringc.wikispaces.com/Sm%C3%A5+Opgaver>
Den metode der bliver brugt til at blande i opgave 4, er også den jeg anvender til at blande kortne. Sidst set d. 24/04 2014.
- http://s3.amazonaws.com/cah/CAH_MainGame.pdf
De officielle regler for spillet. Sidst set d. 24/04 2014.
- <http://mrbool.com/reading-excel-file-with-java/24562>
En måde at læse excel filer i java. Bliver ikke anvendt i spillet. Sidst set d. 25/04 2014.
- <http://stackoverflow.com/questions/3853628/>
En måde at importere .txt filer i java. Sidst set d. 24/04 2014.

9 Billag

9.1 Bruger historier

9.1.1 Brugerhistorie 1 - Antal spillere

Denne brugerhistorie indeholder valget af hvor mange spillere der er i et givent spil. Den indeholder aktørende Spillet og Spillerne.

Forudsætningerne for brugerhistorien er at spillet lige er startet og at der ikke er foretaget nogle valg endnu.

1. Programmet præsenterer spillerne med en skærm med overskrift (Cards Against Humanity) og en dropdown hvor de kan vælge hvor mange spillere de er.
2. De kan vælge mellem 4-5-6-7-8.
3. Spillerne vælger det antal de er og trykker på videre.
4. Spillet opdaterer side 2 så den passer med deres valgte antal og skifter dem over til denne side.

Afslutning: Spillerne har valgt hvor mange de er og er nu klar til at forsætte.

9.1.2 Brugerhistorie 2 – Oprettelse af spillerne

Denne brugerhistorie omhandler at angive navnet på de spillere der er i spillet. Den indeholder aktørende Spillet og Spillerne.

Forudsætningerne for denne brugerhistorie er at spillerne har valgt at de er 7 og trykket videre. Programmet har opdateret side 2 til at der er 7 der kan skrive deres navn

1. Programmet præsenterer spillerne med den skærm hvor de kan skrive deres 7 navn.
2. Spillerne skriver hver deres navn
3. Spillerne trykke videre
4. Spillet oprette alle spillerne

Afslutning: spillet har oprettet alle spillerne og er klar til at gå videre

9.1.3 Brugerhistorie 3 – Kortene

Denne brugerhistorie omhandler at spillet opretter de 2 dæk som der skal bruges til at spille. Den indeholder aktøren spillet

Forudsætninger: Spillerne har skrevet deres navn og valg hvor mange de er og trykket videre.

1. Spillet opretter 2 lister, en til normale kort og en til de sorte kort
2. Spillet opretter de kort der skal være i hver af de 2 lister.
3. Spillet blander kortene

Afslutning: Spillet er klar til at gå i gang med at spille.

9.1.4 Brugerhistorie 4 – En runde

Denne brugerhistorie omhandler en hel runde af spillet. Den indeholder aktørende spillet og spiller 1-4.

Forudsætning: Spillerne har valgt at de er 4 spiller og indtastet deres navn. De har også trykket viddere.

1. Spillet deler nu 10 kort ud til hvert spiller.
2. Der dukker en besked op på skærmen om at det navn spiller 1 har valgt skal læse et sort kort op.
3. Spiller 1 læser kortet op og trykker næste, spillet siger nu at det er det navn spiller 2 har valgt som der skal vælge hvilket svar han/hun vil give
4. Han trykker på det kort han vil aflevere. Han trykker på næste, og turen går nu viddere til spiller 3.
5. Dette gentages indtil alle spillere, undtagen ham/hun der læste et kort op, har valgt et kort.
6. Den spiller der læste et kort op får nu en skærm hvor at han/hun kan vælge hvilket kort han/hun synes passer best. Han/hun klikker så på dette.
7. Den spiller kortet tilhøre får et point.
8. Alle spillere undtagen ham/hun der læste op får nu et ekstra kort.

Afslutning: runden er slut, og der startes nu en ny runde. Den spiller der kom efter ham/hun skal nu læse et kort op.

9.1.5 Brugerhistorie 5 – Afslutningen af et spil

Denne brugerhistorie omhandler afslutningen af et spil. Den indeholder spillerne og den indeholder MainClasse.

Forudsætning: Spillerne har spillet en masse runder indtil der enten ikke er flere kort, eller de ikke gidder spille mere.

1. Spillerne klikker på afslutspillet, eller hvis der ikke er flere kort sker det af sig selv.
2. Spillet viser en skærm hvor man kan se hvor mange points hver spiller har fået, og hvem der har fået flest.
3. Spillerne kan trykke "afslut spil" eller "spil igen".

Afslutning: spillerne trykker afslut spil eller spil igen. Hvis de trykker afslut spil lukker spillet ned. Hvis de trykke spil igen, se den brugerhistorie.

9.1.6 Brugerhistorie 6 – Spil igen

Denne brugerhistorie omhandler afslutningen af et spil. Den indeholder spillerne og den indeholder MainClasse.

Forudsætninger: spillerne har set hvem der har vundet og er klar til at spille igen. De er præsenteret med en skærm hvor de kan vælge i mellem at afslutte spillet eller at spille igen.

1. Spillerne vælger spil igen
2. Spillet nulstiller alle variabler.
3. Spillet diregerre dem nu til start skærmen, hvor de kan vælge hvor mang espiller de er.

Afslutning: spillet er klart til at de kan spille igen.

9.2 Kodens

9.2.1 Den primære klasse - MainClass

```
1  /*
2  Copyright (C) 2014 Simon Skjernaa Erfurth
3
4
```

5 Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

6
7 The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

8
9 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.

10 */
11
12 import java.awt.EventQueue;
13
14 import javax.swing.JFrame;
15 import javax.swing.JPanel;
16
17 import java.awt.CardLayout;
18
19 import javax.swing.JLabel;
20
21 import java.awt.Font;
22 import java.awt.Color;
23
24 import javax.swing.JComboBox;

```
25 import javax.swing.JButton;
26
27 import java.awt.event.ActionListener;
28 import java.awt.event.ActionEvent;
29
30 import javax.swing.JTextField;
31
32
33 public class MainClass {
34
35     //Vi opretter allerede nu en række variabler og lister som
36     //vi ønsker at have adgang til i gennem hele spillet.
37     JComboBox antalSpiller = new JComboBox();
38     JPanel secondpage = new JPanel();
39     Player[] spillere;
40     Cards[] playdeck;
41     Cards[] blackdeck;
42     JTextField[] txtName;
43     int antalspillere;
44     CardLayout c; //Denne bliver brugt for at faa vores
45     //CardLayout til at fungere ordentligt.
46
47     private JFrame frame;
48
49     /**
50      * Launch the application.
51      */
52     public static void main(String[] args) {
53         EventQueue.invokeLater(new Runnable() {
54             public void run() {
55                 try {
56                     MainClass window = new
57                         MainClass();
58                     window.frame.setVisible(true);
59                 } catch (Exception e) {
```

```
57         e.printStackTrace();
58     }
59 }
60 });
61 }
62
63 /**
64  * Create the application.
65  */
66 public MainClass() {
67     initialize();
68     //ET SPIL SKREVET AF SIMON SKJERNAA ERFURTH
69 }
70
71 /**
72  * Initialize the contents of the frame.
73  */
74
75 private void initialize() {
76     txtName = new JTextField[9];
77     frame = new JFrame();
78     frame.setBounds(100, 100, 900, 600);
79     //Vi har valgt at det skal starte i et punkt paa
80     100,100 hvilket betyder at den ikke vil ramme
81     ind i startmenuen, selvom man har startbaren
82     liggende lodret.
83     //Vi har saa valgt oploesningen 900x600 da den
84     sagtens kan vaere paa de fleste skaerme.
85     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
86     frame.setTitle("Cards Against Humanity");
87     //Vi vaelger naturligvis ogsaa at kalde vinduet
88     "Cards agaisnt humanity", saa det er let at
89     identificere.
```

```
85         frame.getContentPane().setLayout(new CardLayout(0,
86             0));
87         //Her vaelger vi at vinduet skal bruge et card
88         layout.
89         //Start siden, vaelg antal spillere
90         JPanel firstpage = new JPanel();
91         frame.getContentPane().add(firstpage, "Start");
92         firstpage.setLayout(null);
93         //Side to, navngiv spillerne
94         frame.getContentPane().add(secondpage,
95             "Navngivning");
96         //Side 3, spil 1
97         JPanel play1 = new JPanel();
98         frame.getContentPane().add(play1, "Laes");
99
100         //Side 4, spil 2 (Blev aldrig brugt)
101         JPanel play2 = new JPanel();
102         frame.getContentPane().add(play2, "Spil");
103
104         //Side 5, spil 3 (Blev aldrig brugt)
105         JPanel play3 = new JPanel();
106         frame.getContentPane().add(play3, "Vaelg");
107
108         //Side 6, scoreboard (Blev aldrig brugt)
109         JPanel score = new JPanel();
110         frame.getContentPane().add(score, "Scoreboard");
111
112         //-----
113         //DELE TIL FIRSTPAGE
114         //-----
115
```

```
116         JLabel lblWelcome = new JLabel("CARDS AGAINST  
            HUMANITY");  
117         //Vi laver her en overskrift saa man kan se hvad man  
            har startet.  
118         lblWelcome.setBounds(115, 5, 654, 63);  
119         lblWelcome.setForeground(Color.DARK_GRAY);  
120         lblWelcome.setFont(new Font("Tahoma", Font.PLAIN,  
            52));  
121         firstpage.add(lblWelcome);  
122  
123         antalSpiller.setMaximumRowCount(5);  
124         antalSpiller.setBounds(350, 100, 200, 50);  
125         antalSpiller.addItem(4);  
126         antalSpiller.addItem(5);  
127         antalSpiller.addItem(6);  
128         antalSpiller.addItem(7);  
129         antalSpiller.addItem(8);  
130         firstpage.add(antalSpiller);  
131         //Her laver vi en combobox, saa man kan vaelge hvor  
            mange spillere man er.  
132  
133         JLabel lblAntalSpillere = new JLabel("Antal  
            spillere:");  
134         lblAntalSpillere.setBounds(260, 118, 80, 14);  
135         firstpage.add(lblAntalSpillere);  
136  
137         //Vi laver en knap til at gaa til naeste side, denne  
            skifter cardlayoutet, opdatere naeste side og  
            saetter listen spillere til at have den  
            stoerrelse som passer med hvor mange spillere  
            man har valgt man er.  
138         JButton btnNewButton = new JButton("Next");  
139         btnNewButton.addActionListener(new ActionListener() {  
140             public void actionPerformed(ActionEvent arg0)  
            {
```

```
141         antalspillere =
            antalSpiller.getSelectedIndex() +
            4 ;
142         c =
            (CardLayout)frame.getContentPane().getLayout();
143         c.show(frame.getContentPane(),
            "Navngivning");
144         opdaterSide2();
145         spillere = new Player[antalspillere+1];
146
147     }
148 });
149 btnNewButton.setFont(new Font("Tahoma", Font.PLAIN,
            18));
150 btnNewButton.setBounds(697, 500, 177, 50);
151 firstpage.add(btnNewButton);
152 secondpage.setLayout(null);
153
154 //-----
155 //DELE TIL SECONDPAGE
156 //-----
157
158 JLabel lblWelcome2 = new JLabel("CARDS AGAINST
            HUMANITY");
159 lblWelcome2.setBounds(115, 5, 654, 63);
160 lblWelcome2.setForeground(Color.DARK_GRAY);
161 lblWelcome2.setFont(new Font("Tahoma", Font.PLAIN,
            52));
162 secondpage.add(lblWelcome2);
163
164 //Dette kunne veare lavet med en "for" saetning med
            fordel, men blev af tidsmaessige aarsager ikke
            lavet saadan.
165 JLabel lblPlayer1 = new JLabel("Player 1:");
166 lblPlayer1.setBounds(36, 110, 57, 14);
```



```
167         secondpage.add(lblPlayer1);
168
169         txtName[1] = new JTextField();
170         txtName[1].setText("Name");
171         txtName[1].setBounds(101, 107, 86, 20);
172         secondpage.add(txtName[1]);
173
174         txtName[2] = new JTextField();
175         txtName[2].setText("Name");
176         txtName[2].setColumns(10);
177         txtName[2].setBounds(320, 107, 86, 20);
178         secondpage.add(txtName[2]);
179
180         JLabel lblPlayer2 = new JLabel("Player 2:");
181         lblPlayer2.setBounds(246, 113, 66, 14);
182         secondpage.add(lblPlayer2);
183
184         txtName[3] = new JTextField();
185         txtName[3].setText("Name");
186         txtName[3].setColumns(10);
187         txtName[3].setBounds(526, 107, 86, 20);
188         secondpage.add(txtName[3]);
189
190         JLabel lblPlayer3 = new JLabel("Player 3:");
191         lblPlayer3.setBounds(453, 113, 66, 14);
192         secondpage.add(lblPlayer3);
193
194         txtName[4] = new JTextField();
195         txtName[4].setText("Name");
196         txtName[4].setColumns(10);
197         txtName[4].setBounds(766, 107, 86, 20);
198         secondpage.add(txtName[4]);
199
200         JLabel lblPlayer4 = new JLabel("Player 4:");
201         lblPlayer4.setBounds(693, 113, 66, 14);
```

```
202         secondpage.add(lblPlayer4);
203
204         //Her laver vi egne en naeste kanp, som skifter til
                naeste side. Den opretter ogsaa daekkene og
                spillerne.
205         JButton lblNext2 = new JButton("Next");
206         lblNext2.addActionListener(new ActionListener() {
207             public void actionPerformed(ActionEvent arg0)
                {
208                 Creatdeck();
209                 for(int ii= 1; ii < antalspillere +1;
                        ii++){
210                     spillere[ii] = new Player(ii,
                            txtName[ii]);
211                     //System.out.println("spler" +
                            spillere[ii].id + "hedder"
                            + spillere[ii].name + " og
                            har " + spillere[ii].points
                            + "points");
212                     //Denne linje var for at teste
                            at der korrekt blev
                            oprettet spillerne i listen.
213                 }
214
215                 c =
                        (CardLayout)frame.getContentPane().getLayout();
216                 c.show(frame.getContentPane(), "Laes");
217             }
218         });
219         lblNext2.setFont(new Font("Tahoma", Font.PLAIN, 18));
220         lblNext2.setBounds(675, 481, 177, 50);
221         secondpage.add(lblNext2);
222
223         //SIDE 3 - Her skal den sorte spiller laese et kort
                hoejt
```

```
224
225         //Side 4 - Her skal de andre spillere vælge hvilket
           kort de vil spille
226
227         //Side 5 - Her skal den sorte spiller vælge hvilket
           kort han ønsker skal vinde
228
229         //SIDE 6 - HER SKAL POINT STILLINGEN OSV. FREMGAA.
230
231
232     }
233
234     protected void Creatdeck() {
235         //Denne metode oprette de 2 dæk vi anvender og
           blander dem.
236
237         //*****
           //Dette er det næste der ville være blevet lavet
           færdigt i projektet.
           //*****
238
239
240         playdeck = new Cards[10];
241         //playdeck er de hvide kort som spillerne ville
           skulle spille med. Dette er ikke som det
           endeligt ville have været.
242         for(int ii =0; ii <10;ii++){
243             playdeck[ii]= new Cards(ii, "Svar nummer " +
                ii + ".");
244             //System.out.println("Kort nummer " +
                playdeck[ii].id + " har teksten " +
                playdeck[ii].text +".");
245             //Ovenstaaende er et debugging værktøj til
                at teste at de bliver oprettet.
246         }
247
248         blackdeck = new Cards[10];
```

```
249         //blackdeck er de sorte kort som spilelrne skal
           laese hoejt. Dette er ikke som det endeligt
           ville have vaeret.
250     for(int ii =0; ii <10;ii++){
251         blackdeck[ii]= new Cards(ii, "Spoergsmaal
           nummer " + ii + ".");
252         //System.out.println("Kort nummer " +
           blackdeck[ii].id + " har teksten " +
           blackdeck[ii].text);
253         //Ovenstaaende er et debugging vaerktoej til
           at teste at de bliver oprettet.
254     }
255
256     //Blanding af listerne. Laant fra
           http://programmingc.wikispaces.com/Sm%C3%A5+Opgaver
257     for(int i = 9; i > 1; i--){
258         //Foerst vaelges en tilfaeldig plads i
           listen
259         int j = (int)(Math.random()*(i+1));
260
261         //Saa byttes tallet vi er naaet til, med
           den tilfaeldige plads i listen.
262         Cards tmp = playdeck[j];
263         playdeck[j] = playdeck[i];
264         playdeck[i] = tmp;
265     }
266
267     for(int i = 9; i > 1; i--){
268         //Foerst vaelges en tilfaeldig plads i
           listen
269         int j = (int)(Math.random()*(i+1));
270
271         //Saa byttes tallet vi er naaet til, med
           den tilfaeldige plads i listen.
272         Cards tmp = blackdeck[j];
```

```
273         blackdeck[j] = blackdeck[i];
274         blackdeck[i] = tmp;
275     }
276
277 }
278
279 protected void opdaterSide2() {
280     //Her opdateres den 2. side til at indeholde det korrekte
281     //antal felter og labels. Dette kunne være lavet mere
282     //elegant med en "for" sætning.
283     if(antalspillere>4){
284         JLabel lblPlayer5 = new JLabel("Player 5:");
285         lblPlayer5.setBounds(36, 318, 57, 14);
286         secondpage.add(lblPlayer5);
287
288         txtName[5] = new JTextField();
289         txtName[5].setText("Name");
290         txtName[5].setColumns(10);
291         txtName[5].setBounds(101, 315, 86, 20);
292         secondpage.add(txtName[5]);
293     }
294
295     if(antalspillere>5){
296         txtName[6] = new JTextField();
297         txtName[6].setText("Name");
298         txtName[6].setColumns(10);
299         txtName[6].setBounds(320, 315, 86, 20);
300         secondpage.add(txtName[6]);
301
302         JLabel lblPlayer6 = new JLabel("Player 6:");
303         lblPlayer6.setBounds(246, 321, 66, 14);
304         secondpage.add(lblPlayer6);
305     }
306
307     if(antalspillere>6){
```

```
306         txtName[7] = new JTextField();
307         txtName[7].setText("Name");
308         txtName[7].setColumns(10);
309         txtName[7].setBounds(526, 315, 86, 20);
310         secondpage.add(txtName[7]);
311
312         JLabel lblPlayer7 = new JLabel("Player 7:");
313         lblPlayer7.setBounds(453, 321, 66, 14);
314         secondpage.add(lblPlayer7);
315     }
316
317     if(antalspillere>7){
318         txtName[8] = new JTextField();
319         txtName[8].setText("Name");
320         txtName[8].setColumns(10);
321         txtName[8].setBounds(766, 315, 86, 20);
322         secondpage.add(txtName[8]);
323
324         JLabel lblPlayer8 = new JLabel("Player 8:");
325         lblPlayer8.setBounds(693, 321, 66, 14);
326         secondpage.add(lblPlayer8);
327     }
328 }
329 }
```

9.2.2 Data klassen Cards

```
1
2
3  /*
4  Copyright (C) 2014 Simon Skjernaa Erfurth
5
6
```

```
7  Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation
   files (the "Software"), to deal in the Software without
   restriction, including without limitation the rights to use,
   copy, modify, merge, publish, distribute, sublicense, and/or
   sell copies of the Software, and to permit persons to whom the
   Software is furnished to do so, subject to the following
   conditions:
8
9  The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.
10
11 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
   OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
   HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
   WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
   FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
   OTHER DEALINGS IN THE SOFTWARE.
12  */
13
14
15 public class Cards {
16     public int id; //Kortets nummer
17     public String text; //Den tekst der er paa kortet
18     public int status; //Statusen paa kortet
19
20     //Status 0 er i bunken
21     //Status 1 er i haanden paa en spiller
22     //Status 2 er at det er blevet spillet
23
24     //Metoden til at oprette et nyt kort.
25     public Cards(int number, String data){
26         this.id = number;
```

```
27         this.text = data;
28         this.status = 0;
29     }
30 }
```

9.2.3 Data klassen Player

```
1  import javax.swing.JTextField;
2
3  /*
4  Copyright (C) 2014 Simon Skjernaa Erfurth
5
6
7  Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation
   files (the "Software"), to deal in the Software without
   restriction, including without limitation the rights to use,
   copy, modify, merge, publish, distribute, sublicense, and/or
   sell copies of the Software, and to permit persons to whom the
   Software is furnished to do so, subject to the following
   conditions:
8
9  The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.
10
11  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
   OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
   HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
   WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
   FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
   OTHER DEALINGS IN THE SOFTWARE.
12  */
```



```
13
14
15 public class Player {
16     public int id; //Spillerens nummer
17     public String name; //Det navn som spilleren indtastede
18     public int points; //Spillerens points
19
20     //Metoden til at oprette en ny spiller
21     public Player(int number, JTextField navn){
22         String navnet = navn.getText();
23         this.id = number;
24         this.name = navnet;
25         this.points = 0;
26     }
27 }
```
