|  |
| --- |
|  |
| 1. års projekt 2013 |
|  |

|  |
| --- |
| Anette Stidsing, Thomas Nielsen, Niklas Renner Nielsen, og Martin Hana  06-06-2013 |



Indhold

[Fordeling af ansvar 4](#_Toc358275979)

[Afgrænsning 6](#_Toc358275980)

[Systemudvikling 6](#_Toc358275981)

[Analyse af problemområdet 6](#_Toc358275982)

[Klasser 6](#_Toc358275983)

[Klassediagram 9](#_Toc358275984)

[Hændelser 10](#_Toc358275985)

[Hændelsestabel 12](#_Toc358275986)

[Centrale klasser - Adfærdsmønstre beskrevet i tilstandsdiagram 15](#_Toc358275987)

[17](#_Toc358275988)

[Analyse af anvendelsesområdet 18](#_Toc358275989)

[Aktørtabel 18](#_Toc358275990)

[Aktører 18](#_Toc358275991)

[Sekvensdiagram 19](#_Toc358275992)

[Funktionsliste 20](#_Toc358275993)

[Brugergrænsefladen 21](#_Toc358275994)

[Navigationsdiagram 22](#_Toc358275995)

[Designdokumentet 24](#_Toc358275996)

[Opdeling af komponenter, beskrevet i et UML diagram 24](#_Toc358275997)

[Komplekse operationer 27](#_Toc358275998)

[Database 28](#_Toc358275999)

[Fase 1 28](#_Toc358276000)

[Entitetsdiagram 31](#_Toc358276001)

[Fase 2 32](#_Toc358276002)

[Forbindelser 32](#_Toc358276003)

[ER diagram 33](#_Toc358276004)

[Fase 3 34](#_Toc358276005)

[Beskrivelse af alle 7 punkter i konverteringen 34](#_Toc358276006)

[EER diagram 36](#_Toc358276007)

[Fase 4 37](#_Toc358276008)

[Indsæt data 37](#_Toc358276009)

[Fase 5 41](#_Toc358276010)

[Test 41](#_Toc358276011)

[Programmering 43](#_Toc358276012)

[Beskrivelse af program 43](#_Toc358276013)

[Lynsalg 44](#_Toc358276014)

[Lager: 44](#_Toc358276015)

[Ordresalg: 44](#_Toc358276016)

[Klasser og centrale metoder 46](#_Toc358276017)

[DBConnection klassen 46](#_Toc358276018)

[DatabaseObjectHandler klassen 46](#_Toc358276019)

[LynSalg 48](#_Toc358276020)

[Panel\_OrdreSalg og Panel\_OrdreSalgLinje 49](#_Toc358276021)

[Panel\_Lager klassen 52](#_Toc358276022)

[Panel\_LagerLinje klassen 52](#_Toc358276023)

[JDialog\_LagerVare klassen 52](#_Toc358276024)

[Søg 53](#_Toc358276025)

[Fejlhåndtering 55](#_Toc358276026)

[Test 59](#_Toc358276027)

[Testoversigt 59](#_Toc358276028)

[Generelle forudsætninger: 59](#_Toc358276029)

[It i organisationen 60](#_Toc358276030)

[Bejerholm Stenhuggeri ApS, Organisation og it 60](#_Toc358276031)

[Beskrivelse og arbejdsgang 61](#_Toc358276032)

[IT og vækst 63](#_Toc358276033)

[ERP systemet 64](#_Toc358276034)

[Organisation og It-sikkerhed 66](#_Toc358276035)

[Hvorfor har vi valgt: 66](#_Toc358276036)

[IT-Politik. 67](#_Toc358276037)

[Logisk Sikkerhed 67](#_Toc358276038)

[Fysisk sikkerhed 67](#_Toc358276039)

[Datasikkerhed 68](#_Toc358276040)

[Persondataloven 69](#_Toc358276041)

[Slutevaluering 70](#_Toc358276042)

# Fordeling af ansvar

|  |  |
| --- | --- |
| Område | Hovedansvar |
| Systemudvikling - problemområdet |  |
| Klasser | Anette |
| Klassediagram | Thomas, Niklas, Anette og Martin |
| Hændelser og hændelsestabel | Anette |
| Centrale klasser - tilstandsdiagram | Anette |
| Systemudvikling - anvendelsesområdet |  |
| Aktørtabel | Martin |
| Sekvensdiagram | Thomas |
| Funktionsliste | Niklas |
| Brugergrænsefladen | Thomas |
| Systemudvikling - Designdokument |  |
| UML diagram | Anette |
| Komplekse operationer | Niklas |
| Database fase 1 |  |
| Entiteter | Thomas |
| Entitets diagram | Thomas |
| Database fase 2 |  |
| Forbindelser | Anette |
| Er diagram | Anette |
| Database fase 3 |  |
| Konvertering | Niklas |
| EER diagram | Niklas |
| Database fase 4 |  |
| Data | Martin |
| Fase 5 |  |
| Test | Anette |
| Programmering - beskrivelse af program |  |
|  |  |
| Programmering - klasser og centrale metoder |  |
| UML diagram (Henvises til Designdokument afsnit) | Anette |
| Lynsalg | Thomas |
| Lager | Niklas |
| Ordresalg | Anette |
| Login | Thomas |
| Søg | Thomas og Niklas |
| MainGUI | Thomas |
| Objekthandler | Niklas og Anette |
| Connection gui | Martin |
| CSV | Martin |
| Administration | Martin |
| ProvisionsseddelPDF | Anette |
| OrdrePDF | Niklas |
| FakturaPDF | Niklas |
| KontoudtogPDF | Thomas |
| Revisorseddel | Thomas |
| Programmering - fejlhåndtering |  |
|  |  |
| Programmering - Test |  |
| IT i organisationen |  |
| IT og forretning | Anette |
| ERP | Martin |
|  | Niklas |
| Organisation og it sikkerhed | Thomas |

# Afgrænsning

Formålet med dette projekt er, at lave et system til Bejerholm Stenhuggeri, der kan hjælpe ham med at håndtere varelager, samt håndtere salg.

I opgavebeskrivelsen, beder Bejerholm om at

# Systemudvikling

## Analyse af problemområdet

### Klasser

#### Kunde:

Denne klasse skal bruges til registrering af fornavn, efternavn, adresse, postnummer samt tlf der identificerer kunden.

#### Samarbejdspartner

Samarbejdspartner skal registrere firmanavn, adresse, tlf, cvr nr, registrerings- og kontonummer samt bank. Denne klasse identificerer samarbejdspartnere og giver mulighed for at gemme disse.

#### Kirkegård

En kirkegård indeholder et id og et navn. Den har til formål at knyttes sammen med en ordre, hvis en gravsten er valgt.

#### Ordre:

En ordre skal registrere information om ordredato, som er ordrens oprettelsesdato. Den skal samtidig have en leverings- og en afhentningsdato, for at kunne registrere, hvilket tidspunkt en ordre enten skal afhentes eller leveres til kunden. Ordren skal også have en ordretype, idet ordren enten kan indeholde en ny sten og en tilføjelse. Er ordren en tilføjelse, skal den have en afhentningsdato. Er ordretypen derimod en tilføjelse, skal den have en leveringsdato. Ordren indeholder attributter til information til kirkegård(adresse, række, nr, gravtype), hvis ordren er en gravsten. Ordren indeholder en leveringsadresse, men ikke en afhentningsadresse da afhentningsadressen i næsten alle tilfælde vil være en kirkegård. Ordre er altså en central klasse, som skal binde en kunde og varelinje sammen.

#### Vare

Denne klasse skal registrere alle fysiske vare, der er på lageret, og informationer om denne; navn, højde, bredde, indkøbspris, salgspris, overflade, om varen har en dekoration, varens status og et id til identificering.

#### Varetype:

Varetype skal indeholde et id og et navn. Klassen har til formål at bestemme hvilke type en vare er. En vare med en varetype kan f.eks. være en dekoration med typen o. storm og en stenplade kan have varetypen granitplade.

#### Varegruppe

En varegruppe skal bestemme hvilken varegruppe en vare er tilknyttet.

#### Tom\_linje:

En tomlinje skal indeholde navn, pris og antal. Denne kan bruges til at oprette ydelser som rensning, værkstedstimer eller en vare som ikke er på lageret.

#### Inskription

Inskription bruges til at samle inskriptionlinjer, og bestemme skrifttypen.

#### Inskription\_linje

Denne klasse indeholder en inskription, som er teksten på inskriptionlinjen. Derudover linjetype som kan være enten ny, gammel eller en plads til en inskription.

#### Tegntype

Tegntype skal have et navn, id og en pris per tegn. Den har til formål at tilknyttes den samlede Inskription, og dermed bestemme hvilken tegntype inskriptionen har og give den en samlet pris.

#### Vare\_linje:

En varelinje indeholder et linje\_nr og har til formål at tilknytte en Inskription, en vare eller en tomlinje til en ordre. Objekter af denne klasse skal altså kun bruges i denne sammenhæng og levetiden for denne er derfor begrænset.

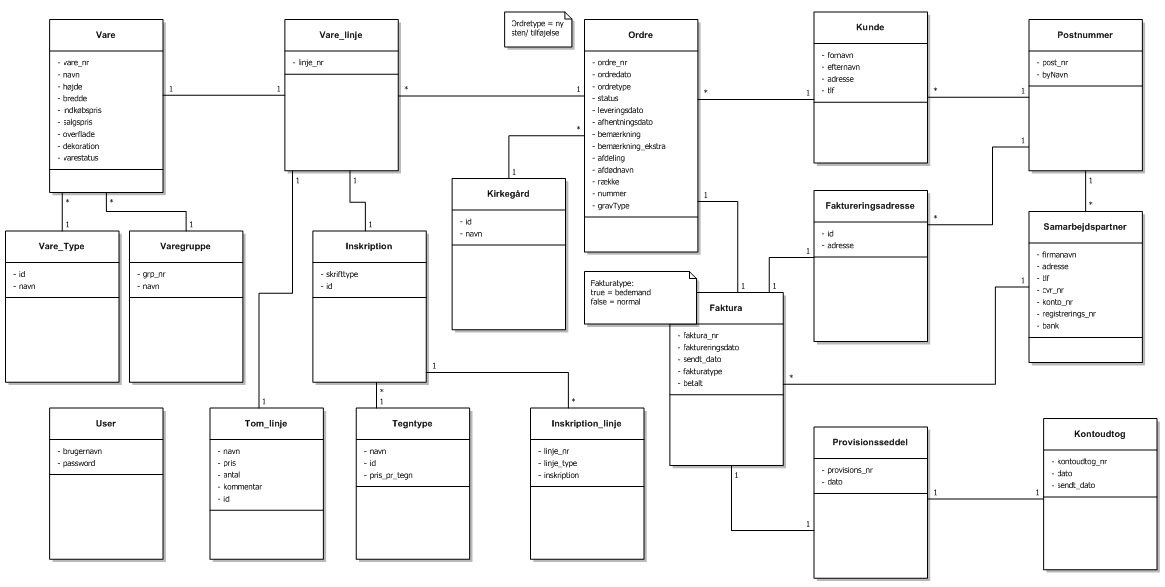
#### Faktura

Faktura er en central klasse, og skal indeholde attributterne faktura\_nr, faktureringsdato, sendt\_dato, fakturatype samt betalt. En fakturatype bestemmer om et faktura objekt skal oprettes til en bedemand eller en privatperson. Hvis fakturatypen er sat til at være til en bedemand skal der oprettes et kontoudtog og en provisionsseddel.

#### Faktureringsadresse

En faktureringsadresse skal have et id og en adresse, og knyttes til en faktura hvis kunden har en anden faktureringsadresse end kundens egen adresse.

### Klassediagram



### Hændelser

#### Kunde, samarbejdspartner, kirkegård, oprettes:

Disse hændelser medtages ikke, idet objekter af klasserne vil blive oprettet idet en ordre bliver oprettet.

#### Der redigeres i kundeinfo, samarbejdspartnerinfo og kirkegårdsinfo:

Idet man kun skal kunne ændre i objekter af klasserne kunde, samarbejdspartner og kirkegård i forbindelse med oprettelse af en ordre eller redigering af en ordre.

#### Kunde, samarbejdspartner, kirkegård slettes:

Disse hændelser skal ikke med, idet det ikke skal være muligt at slette en kunde, en samarbejdspartner eller en kirkegård i programmet.

#### Ordre oprettes:

En central hændelse, der er knyttet til "ordre". Hændelsen involverer også objekter fra klasserne kunde, vare, vare\_linje, Tom\_linje, Tegntype, Inskription og Kirkegård. Idet hændelsen indtræffer, oprettes et "ordre" objekt. Denne hændelse er den første der vedrører en vare\_linje, tom\_linje, tegntype samt inskription, idet disse vil oprettes når denne hændelse sker. En kunde og en kirkegård kan, når denne hændelse indtræffer, både oprettes samt redigeres.

#### Der redigeres i ordre:

Et eksisterende "ordre" objekt, kan blive ændret undervejs. Hændelsen er ligeledes en central hændelse idet den ligesom hændelsen "ordre oprettes" kan vedrøre klasserne kunde, vare\_linje, tom\_linje, tegntype, inskription samt kirkegård.

#### Ordre annulleres:

Idet hændelsen "Ordre oprettes" er medtaget i hændelsestabellen, er det derfor også nødvendigt at registrere en annullering af ordren. Hændelsen er ligeledes central, idet objekter fra klasserne vare\_linje, tom\_linje, tegntype samt inskription vil gå i sluttilstand idet denne hændelse indtræffer. Hændelsen kan forekomme et vilkårligt antal gange for en kunde samt en kirkegård.

#### Ordre opgraderes til faktura:

Dette er en central hændelse som både er knyttet til objekter i klasserne ordre, faktura, vare, faktureringsadresse, samarbejdspartner, provisionsseddel samt kontoudtog. Hændelsen indtræffer idet en ordre er afsluttet og afsendt, og en faktura skal oprettes. Objekter i klasserne "Vare" og "Samarbejdspartner" knyttes til hændelsen, idet et vare objekt ikke længere vil være ledigt i lageret. En samarbejdsparter og en faktureringsadresse kan både oprettes samt redigeres i denne hændelse. Hændelsen er ligeledes den første der vedrører et kontoudtog og et provisionsseddel objekt.

#### Der redigeres i faktura:

En faktura kan ikke blive redigeret idet den oprettes ud fra en ordre. Hændelsen medtages derfor ikke.

#### Der redigeres i faktureringsadresse:

En faktureringsadresse skal ikke kunne redigeres og hændelsen skal derfor ikke med.

#### Ny vare tilføjes på lager:

Denne hændelse er den første der involverer et vare objekt, og tages derfor med i hændelsestabellen.

#### Der redigeres i vareinfo:

Eftersom et vare objekt kan ændres i størrelse, pris mm. er denne hændelse taget med i hændelsestabellen.

#### Vare slettes:

En vare fjernes først fra systemet idet den fjernes manuelt, og hændelsen medtages derfor.

#### Vare er produceret:

Hændelsen er udenfor problemområdets rammer og medtaget derfor ikke i hændelsestabellen.

#### Faktura annulleres og Faktura betalt

En faktura kan gå i sluttilstand ved enten af blive annulleret eller betalt. Hændelsen medtages dermed.

#### Rykker sendes

Hændelsen medtages idet den vedrører en faktura.

#### Ny vare tilføjes på lager:

Dette er den første hændelse, der kan indtræffe for en vare. Derfor medtages hændelsen.

#### Der redigeres i vareinfo:

Denne hændelse vedrører en vare. Den vedrører ligeledes en varetype, idet denne hændelse

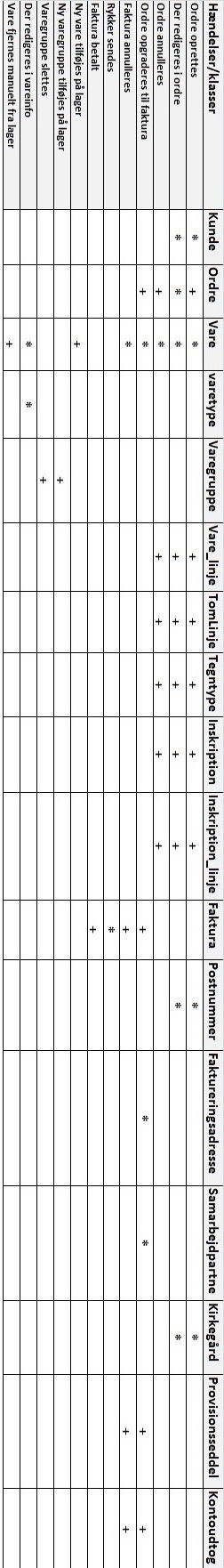
#### Vare fjernes manuelt fra lager:

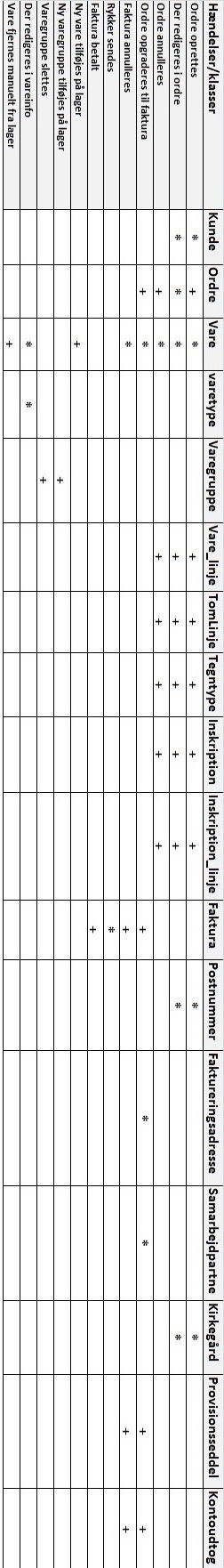
Hændelsen vil føre

#### Provisionsseddel oprettes og kontoudtog oprettes:

Hændelserne medtages ikke, idet disse vil blive oprettet idet en ordre bliver opgraderet til en faktura.

### Hændelsestabel



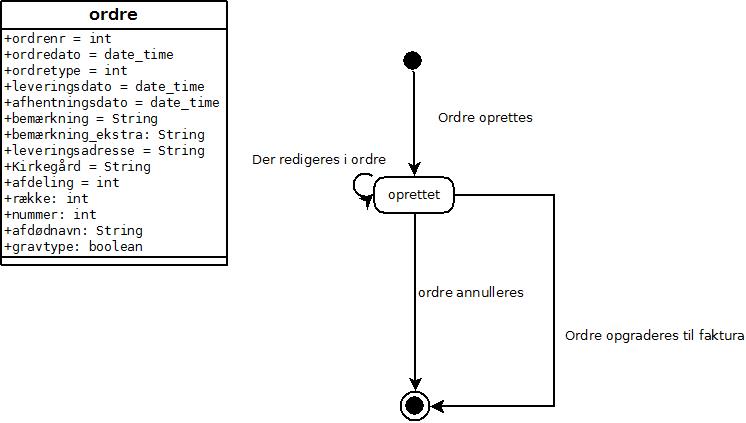


### Centrale klasser - Adfærdsmønstre beskrevet i tilstandsdiagram

#### Ordre

I beskrivelsen af adfærdsmønstret for et ordre objekt, betegner "oprettet", den tilstand en ordre kommer i, idet hændelsen "ordre oprettes" indtræffer. Ordren vil da være aktivt i systemet. I denne tilstand er hændelsen "Der redigeres i ordren" angivet ved en direkte iteration. Hændelsen kan altså indtræffe fra nul til flere gange, og den vil føre direkte tilbage til samme tilstand, som den startede i.

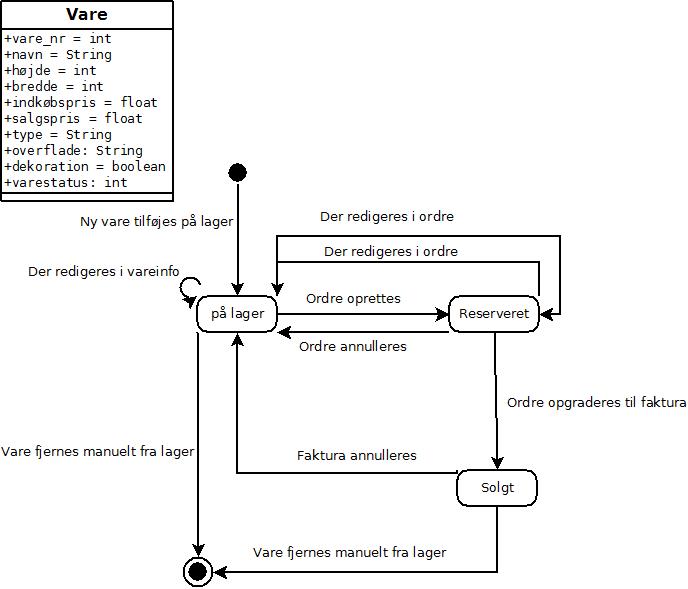
Hændelserne "Ordre opgraderes til faktura" og "ordre annulleres", vil begge føre objektet over i sluttilstanden. Dette er angivet ved en selektion, idet netop en ud af de to mulige hændelser kan indtræffe.

Havde vi valgt at udvide systemet til bedre at kunne håndtere produktionsprocessen, ville tilstandsdiagrammet for et ordre objekt da indeholde "status" som en attribut. Tilstandsdiagrammet for et ordre objekt ville da indeholde tilstandene "modtaget" og "produceret" og et ordre objekt ville dermed kunne ændre tilstand alt efter hvor langt en vare var i produktionsprocessen.

#### Vare

Et vare objekt starter i tilstanden "på lager" idet hændelsen "Ny vare tilføjes på lager" indtræffer. I denne tilstand, er hændelsen "Der redigeres i vareinfo" angivet ved en direkte iteration. En vare kan i alt være i tre tilstande; "på lager", "reserveret" og "solgt".

I tilstanden "på lager" kan hændelserne "ordre oprettes", "ordre redigeres" og "vare fjernes manuelt fra lager" indtræffe. Dette beskrives ved en selektion. Bliver varen fjernet fra lageret, vil den føres til sluttilstanden. Indgår varen derimod i en ordre, vil den ændre tilstand til "reserveret", og der sikres dermed, at der ikke kan opstå dobbeltsalg af varen.

Derfra kan hændelserne "ordre annulleres" og "der redigeres i ordre" indtræffe, og varen vil da blive ført tilbage til den oprindelige tilstand. Dette er angivet ved en indirekte iteration, idet en vare kan gå fra at være "på lager" og værende "reserveret" et vilkårligt antal gange. Er varen "reserveret", kan hændelsen "ordre opgraderes til faktura" indtræffe, og objektet vil derved gå videre i tilstanden "solgt". Da en faktura kan annulleres, vil vare objektet gå tilbage i start tilstanden "på lager", hvis denne hændelse indtræffer, og kun gå i sluttilstand hvis hændelsen "vare fjernes manuelt fra lager", indtræffer.

Idet Bejerholms Stenhuggeri producerer varer efter bestilling, kan virksomheden også beskrives som værende en produktionsvirksomhed. Bestiller en kunde et produkt som ikke findes på lageret, vil dette produceres af de stenplader der eksisterer på varelageret. I et sådan tilfælde sker der altså en forarbejdning af varen, idet en stenplade kan beskrives som værende en råvarer, som bearbejdes til en færdigvare. Eftersom Bejerholm Stenhuggeri er en lille virksomhed, med kun en ansat som både opretter ordre samt producerer varerne er det derfor ikke en nødvendigt funktion i systemet. I stedet fjernes varen manuelt fra lageret, hvis der modtages en ordre på en sten som skal produceres. Den valgte stenplade vil da skulle fjernes fra lageret og derefter tilføjes som en ny vare, med de nye mål og priser på det overskydende som ikke anvendes til produktion.

#### 

#### Faktura

Idet en ordre opgraderes til en faktura, vil der blive oprettet et faktura objekt i tilstanden "ikke betalt". i denne tilstand kan en rykker sendes et vilkårligt antal gange. Varen kan gå i sluttilstand hvis en af de to hændelser "faktura annulleres" eller "faktura betalt" indtræffer. Når en faktura bliver betalt, vil objektet ikke længere være aktivt i systemet og det vil derfor være endeligt.

## C:\Users\Anette Stidsing\Documents\GitHub\Bejerholms-Stenhuggeri\System udvikling\faktura.jpeg

## Analyse af anvendelsesområdet

### Aktørtabel

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Bruger logind/ud | ✓ | ✓ |
| Salg af vare | ✓ | ✓ |
| Varebehandling(opret,redigere,slet) | ✓ |  |
| Varesøgning | ✓ | ✓ |
| Samlet inkøb og salgs for samlede varegrp. Oversigt | ✓ |  |
| Ordre/faktura søgning | ✓ | ✓ |
| Ordrebehandling | ✓ | ✓ |
| Odreopgradering | ✓ | ✓ |
| Provisionssedel og kontoudtogsbehandling | ✓ |  |
| Tjek/ændre faktura og ordrestatus | ✓ | ✓ |
| Dokumentbehandling (email,udprint) | ✓ | ✓ |
| Backupbehandling | ✓ |  |

### Aktører

***Oversigt:*** Vi har to aktører – Bruger og Sælger - og ialt 12 brugsmønstre. Nedenfor beskrives rollerne for systemets to aktører

***Bruger***

**Formål:** Bruger skal have administrativt adgang til systemmet, dvs hvis der skal systemmet er ikke begrænset for Bruger/administrator.

**Karakteristik:** En bruger skal kunne tage backup af kunder, og kunne behandle varene, fylde på, rette eller slette. Derudover skal han kunne oprette en eller flere sælgere til systemet.

**Sælger**

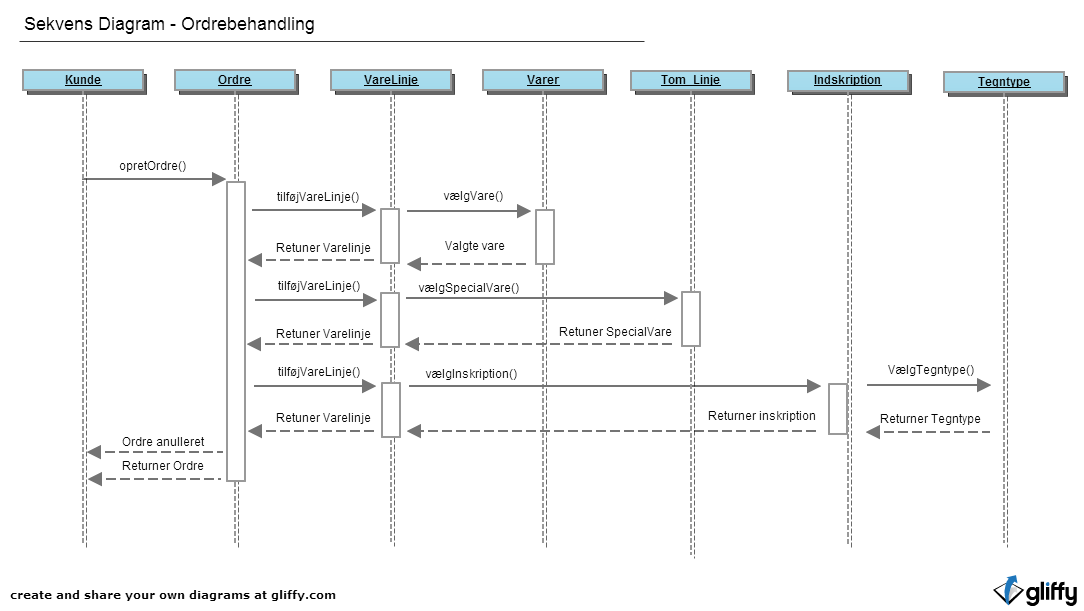
**Formål:** Sælger skal kunne sælge vare og have mindre ansvar for administrativt arbejde i systemet.

**Karakteristik:** En sælger er til for at sælge vare og hjælpe med det grundlæggende i systemet.

### Sekvensdiagram

## Sekvens diagram over brugsmønstret: Ordrebehandling.

Diagrammet viser interaktionen mellem de forskellige klasser ved oprettelse af en ordre. Når en aktør har oprettet en kunde og ønsker at oprette en ordre til denne, kaldes metoden opretOrdre(), som opretter et nyt ordre objekt. I ordre klassen kaldes metoden tilføjVareLinje() som opretter et nyt VareLinje objekt. Dette kan indeholde en vare, speciallinje eller et inskriptions objekt. Der kan kun være et af de før nævnte objekter tilknyttet et objekt af VareLinje, dette ses i diagrammet ved at der oprettes et nyt objekt af varelinje ud fra hvert kald til de 3 klasser. Ønskes der flere ting på ordren oprettes de enkeltvis som en ny varelinje. Ordre objektet lever indtil ordren enten bliver annulleret eller bliver godkendt og returneret til kunde objektet.



## Ændringer

Metoderne angivet i sekvensdiagrammet ovenfor er ikke eksisterende i systemet. Vi valgte at kalde objekternes konstructor direkte i gui’en i stedet. Dvs. når en ordre oprettes kalder vi i gui’en, at der skal oprettes et ordre objekt med et kunde objekt samt en ArrayList ved Varelinjer og så fremdeles.

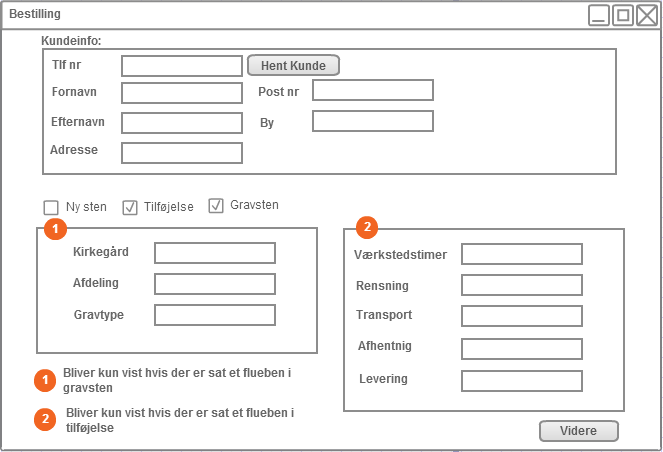
### Funktionsliste

|  |  |  |
| --- | --- | --- |
| Funktionsliste | | |
| Funktion | **Kompleksitet** | **Type** |
| Opret, rediger, fjern vare | Simpel | Opdatering |
| Se samlet købs- og salgspris for en varegruppe | Medium | Beregning |
| Lav kontantsalg | Medium | Beregning |
| Opret, rediger, fjern ordre | Medium | Opdatering |
| Opret ny ordrelinje | Medium | Opdatering |
| Opgrader til faktura | Simpel | Opdatering |
| Opret provisionssedler og kontoudtog | Medium | Opdatering |
| Tjekke/ændre faktura- og ordrestatus | Simpel | Opdatering |
| Email faktura/ordre/kontoudtog/provisionsseddel | Kompleks | Beregning |
| Eksporter kommaseparerede filer | Kompleks | Beregning |
| Opret, rediger, slet bruger | Simpel | Opdatering |
| Opret PDF af ordre/faktura/kontoudtog/provisionsseddel | Kompleks | Beregning |
| Udregn priser på ordrer(total, total+moms) | Kompleks | Beregning |

# Brugergrænseflade:

Til brugergrænsefladen har vi valgt at sætte fokus på, at systemet skal bruges af en aktør, som ikke har de store it-færdigheder.   
Derfor har et af de vigtigste punkter i opbygningen af brugergrænsefladen været at sørge for, at brugeren nemt kan navigere rundt i menu strukturen og, at elementerne i brugergrænsefladen er beskrevet i sådan grad, at brugeren ikke er i tvivl om deres funktioner. Samtidig har vi ved skemaudfyldelse valgt, at brugeren kun skal stilles over for relevante input felter som har en betydning i arbejdet der udføres.

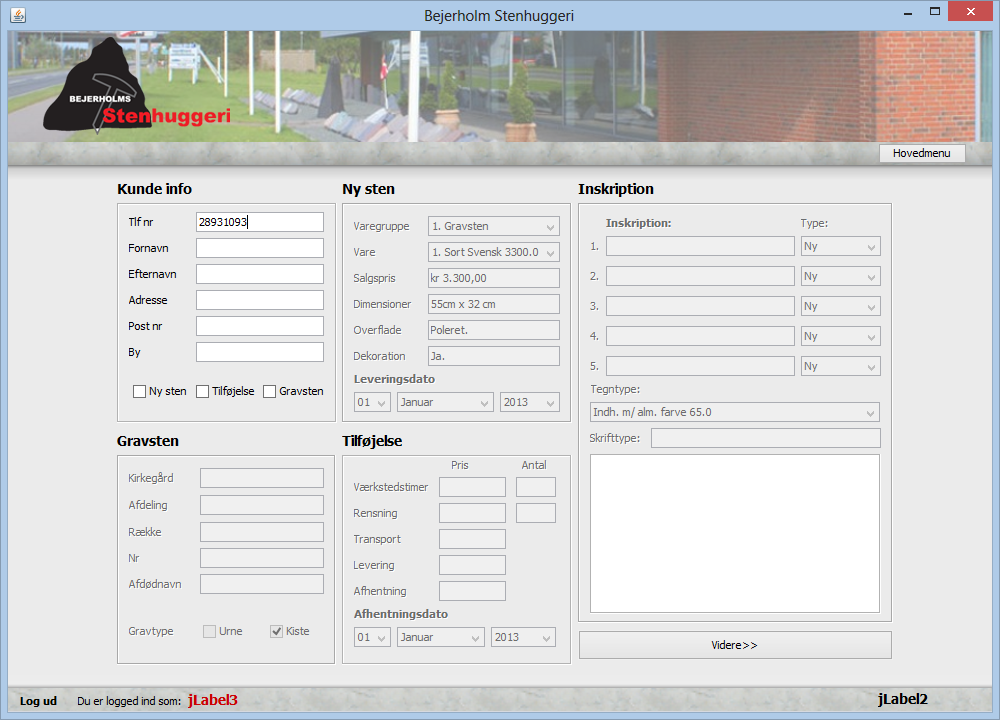
Et eksempel på skemaudfyldelsen kan ses i nedstående skitse over Bestillingsvinduet.



Ydermere har vi for at brugeren nemmere kan bevare overblikket valgt, at man trinvis udfylder ordren. Så man starter med at udfylde informationer om kunden samt ordren, hvorefter ved klik på videre kommer man til hvor man kan tilføje vare ting ordren. Dette er tydeliggjort i navigationsdiagrammet.

## Ændringer.

Skitser over ordresalg viste sig ikke at fyldestgørende nok. Vi havde undervurderet skemaudfyldelsen i sådan grad, at vi drastisk måtte ændre designet på denne. I stedet for at skjule ikke relevante bokse, så bliver de i stedet blot slået fra, så det ikke er muligt at indtaste noget data i dem. Alt efter hvilke checkbokse der bliver markeret, bliver de relevante paneler med input felter låst op. Samtidig er der kommet langt flere input felter ind på siden, der er fordi det er oplysninger der er nødvendige at vide inde brugeren kommer til næste skridt i processen. Det var en ting som vi ikke have taget højde for i den originale skitse.

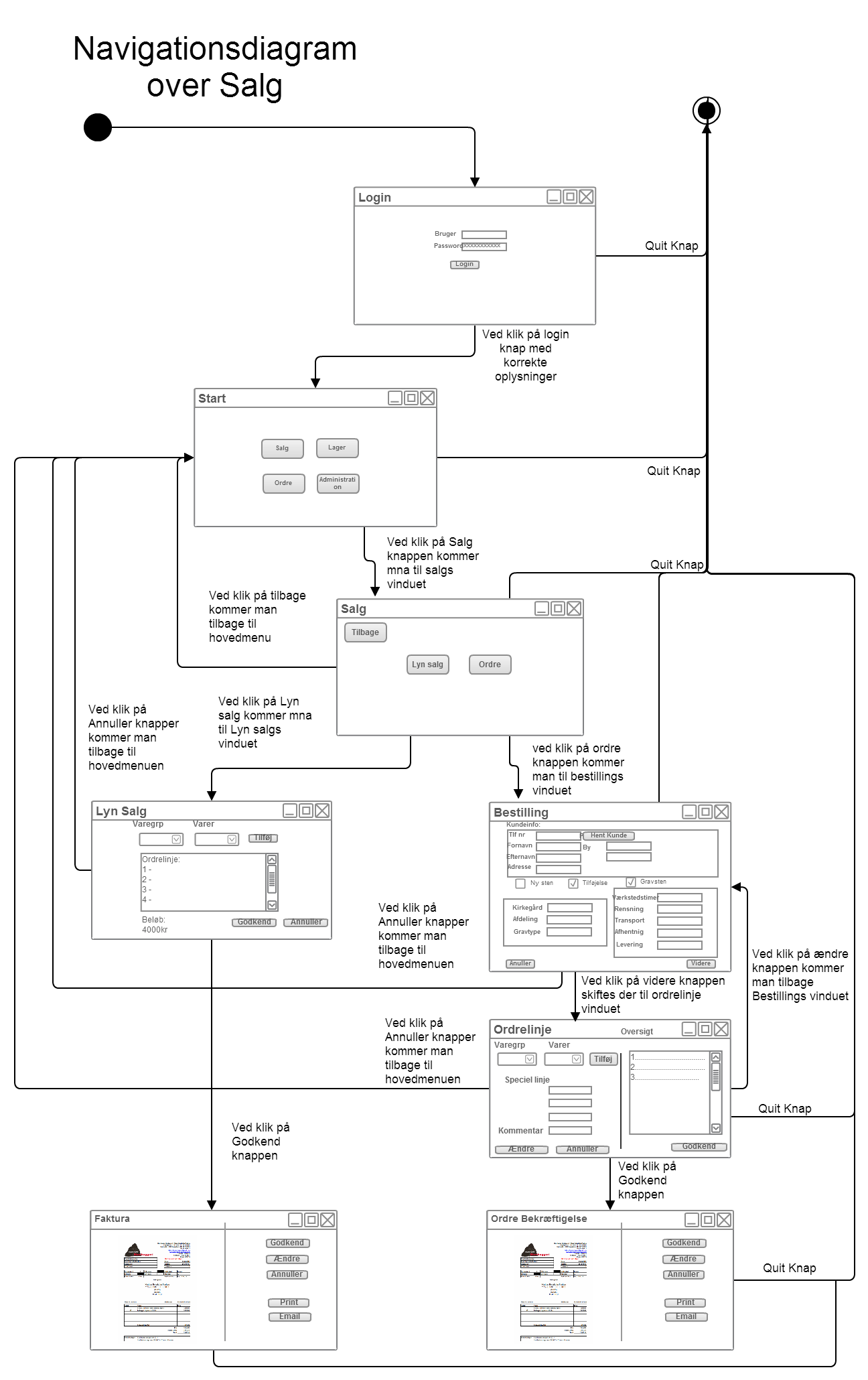


Navigationsdiagram:

Vi har valgt at lave et navigationsdiagram over salg, der viser navigationen i programmet fra det åbnes til salget er udført. Hele omdrejningspunktet i systemet er salgsmenuen, det er derfor vigtigt at gøre denne del så brugervenlig som overhoved mulig. Vi har valgt at dele salg op i 2 underpunkter

* Lyn Salg: Denne bruges ved kontantsalg, altså hvor kunden kommer ned i butikken og vedkommende vælger en eller flere varer fra lageret, som betales og udleveres med det samme i butikken. Der bliver ikke gemt nogen informationer om kunden, vedkommende får bare en faktura med som bevis for betaling.
* Salg: Denne bruges når der er noget arbejde forbundet med salget. Der bliver oprettet en ordre og fakturaen oprettes først når man opgrader ordren.

Vi har valgt netop denne løsning så brugeren af systemet ikke skal lave unødvendig skemaudfyldelse samt for at lette overblikket og arbejdsgangen. Havde vi haft begge dele samlet under et menupunkt ville der være alt for mange data, som slet ikke ville være nødvendige at udfylde, hvis kunden blot ønskede et fuglebad. Det er illustreret i navigationsdiagrammet nedenfor. Ved opdeling spares et helt vindue ved Lyn salg som følge af vores opdeling



## Designdokumentet

### Opdeling af komponenter, beskrevet i et UML diagram







### Komplekse operationer

#### Opret, rediger, fjern vare

Når vi skal oprette en vare i systemet skal vi den have bruger input om alle de informationer varen indeholder. Alle disse input skal valideres så de passer til de pågældende attributter. Derefter skal vi sende alt denne data til databasen, for at oprette varen. Det samme gælder rediger vare, men i stedet for at man skal indsætte alt data, vil der i dette tilfælde være informationer om varen der allerede findes og skal indtastes i programmet, hvorefter kunden kan ændre på disse, som så skal valideres igen og sættes ind i databasen. Fjern vare er simpel da man bare sletter alt vareinformation fra en given vare.

#### Opret, rediger, fjern ordre

Når vi opretter en ordre i system skal den have en masse inputs fra systemet og bruge flere forskellige objekter, som Kunde og Vare\_linjer(Som igen også har mange objekter under sig), alt dette skal valideres i programmet før vi kan sætte det ind i databasen. Vi bliver også nødt til at sætte det ind af flere omgange da et objekt som kunde først skal oprettes før man kan referere til det i ordre. Det samme gælder rediger ordre, bortset fra at vi først skal indsætte den gamle data i programmet hvorefter bruger kan ændre i det, før det valideres og sættes ind i databasen igen. Fjern ordre er også kompleks da den både skal slette Vare\_linjer og under dem også slette inskription, tom\_linje eller sætte en varestatus til at være på lager igen.

#### Eksporter kommaseparerede filer

Denne metode skal først finde alle tabeller i databasen, hvorefter den for hvert enkelte tabel henter alle rows i tabellen, sætter komma mellem hver attribut og linjeskift mellem hver row. Til sidst gemmes alle tabeller som CSV filer i en mappe vi vælger(denne mappe skal også oprettes hvis den ikke findes).

#### Opret PDF af ordre/faktura/kontoudtog/provisionsseddel

De enkelte typer af PDF’er skal oprettes med et objekt der passer til til PDF’en, f.eks. skal en faktura oprettes ved hjælp af et faktura objekt. Når vi så har det rigtige objekt skal vi tegne hele PDF’en og indsætte alle de rigtige data, både standard data som Bejerholms kontaktinfo og logo, samt data fra objektet som vare\_linjer og priser. Til sidst generes PDF’en så og gemmes et sted på harddisken i en mappe vi vælger(mappen skal oprettes hvis den ikke findes i forvejen).

#### Udregn priser på ordrer(total, total+moms)

Da vore Vare\_Linje er kan indeholde en af 3 forskellige objekter skal vi først tjekke om den enkelte Vare\_Linje er en Inskription, en vare eller en tom\_linje. Hvis det er en vare kan vi bare hive salgsprisen fra vare objektet og sætte antal til 1. Hvis det er en inskription skal vi tælle alle nye tegn uden mellemrum og sætte det som antal, og sætte pris\_pr\_tegn i tegntype til enhedspris. Til sidste skal vi så gange enhedspris med antal og så har vi prisen for inskriptionen. Hvis det er en tom \_linje har vi attributter til enhedspris og antal som vi bare kan gange for at få prisen. Til sidst tæller vi så alle vare\_linjernes pris sammen og så har vi totalen uden miljøafgift. Så skal vi lægge 2,5% til og så har vi totalen med miljøafgift. Til sidst ligger vi så 25% til som er moms og så har vi den endelige total. Alle disse tal kan vi så bruge til at udskive priser i vores Program og på vores PDF’er.

# Database

## Fase 1

#### Udskrift af SQL-scriptet 1\_entiteter

drop database if exists bejerholmstenhuggeri;

create database bejerholmstenhuggeri;

use bejerholmstenhuggeri;

drop table if exists kunde;

create table kunde

(

fornavn varchar(50) not null,

efternavn varchar(50) not null,

adresse varchar(50) not null,

tlf char(8) not null,

primary key(tlf)

) engine = innodb;

drop table if exists postnummer;

create table postnummer

(

post\_nr char(4) not null,

byNavn varchar(25) not null,

primary key(post\_nr)

) engine = innodb;

drop table if exists faktureringsadresse;

create table faktureringsadresse

(

id int auto\_increment,

adresse varchar(50) not null,

primary key(id)

) engine = innodb;

drop table if exists ordre;

create table ordre

(

ordre\_nr char(5) not null,

ordretype boolean not null,

ordredato datetime not null,

leveringdato datetime,

afhentningsdato datetime,

bemærkning text,

bemærkning\_ekstra text,

kirkegård varchar(25),

afdeling int,

afdødnavn varchar(50),

række int,

nummer int,

gravType boolean,

primary key(ordre\_nr)

) engine = innodb;

drop table if exists vare\_linje;

create table vare\_linje

(

linje\_nr int not null

) engine = innodb;

drop table if exists tom\_linje;

create table tom\_linje

(

id int auto\_increment,

navn varchar(50) not null,

pris double not null,

antal int not null,

primary key(id)

)engine = innodb;

drop table if exists vare;

create table vare

(

vare\_nr int auto\_increment not null,

navn varchar(100) not null,

højde int,

bredde int,

indkøbspris double not null,

salgspris double not null,

typenavn varchar(50),

overflade int,

dekoration boolean,

vareStatus int not null,

primary key(vare\_nr)

) engine = innodb;

drop table if exists varetype;

create table varetype

(

id int auto\_increment not null,

navn varchar(50) not null,

primary key(id)

) engine =innodb;

drop table if exists varegruppe;

create table varegruppe

(

grp\_nr int auto\_increment not null,

navn varchar(25) not null,

primary key(grp\_nr)

) engine =innodb;

drop table if exists inskription;

create table inskription

(

id int auto\_increment,

skrifttype varchar(25) not null,

primary key(id)

) engine = innodb;

drop table if exists inskription\_linje;

create table inskription\_linje

(

linje\_nr int not null,

inskription varchar(50),

linje\_type int not null

) engine = innodb;

drop table if exists tegntype;

create table tegntype

(

id int auto\_increment,

navn varchar(50) not null,

pris\_pr\_tegn double not null,

primary key(id)

) engine = innodb;

drop table if exists faktura;

create table faktura

(

faktura\_nr char(16) not null,

faktureringsdato datetime,

sendt\_dato datetime,

fakturatype boolean not null,

betalingsstatus boolean,

primary key(faktura\_nr)

) engine = innodb;

drop table if exists kontoudtog;

create table kontoudtog

(

kontoudtog\_nr char(5) not null,

dato datetime,

sendt\_dato datetime,

primary key(kontoudtog\_nr)

)engine = innodb;

drop table if exists provisionsseddel;

create table provisionsseddel

(

provisions\_nr char(5) not null,

dato datetime,

primary key(provisions\_nr)

) engine = innodb;

drop table if exists kirkegård;

create table kirkegård

(

id int auto\_increment not null,

navn varchar(50) not null,

primary key(id)

) engine =innodb;

drop table if exists samarbejdspartner;

create table samarbejdspartner

(

firmanavn varchar(50) not null,

adresse varchar(50) not null,

tlf char(8) not null, # 8 cifre uden landekode

cvr\_nr char(8), # Et CVR nummer er på 8 cifre

registrerings\_nr char(4),

konto\_nr char(10), # Reg + Kontonr

bank varchar(20),

primary key(tlf)

) engine = innodb;

drop table if exists user;

create table user

(

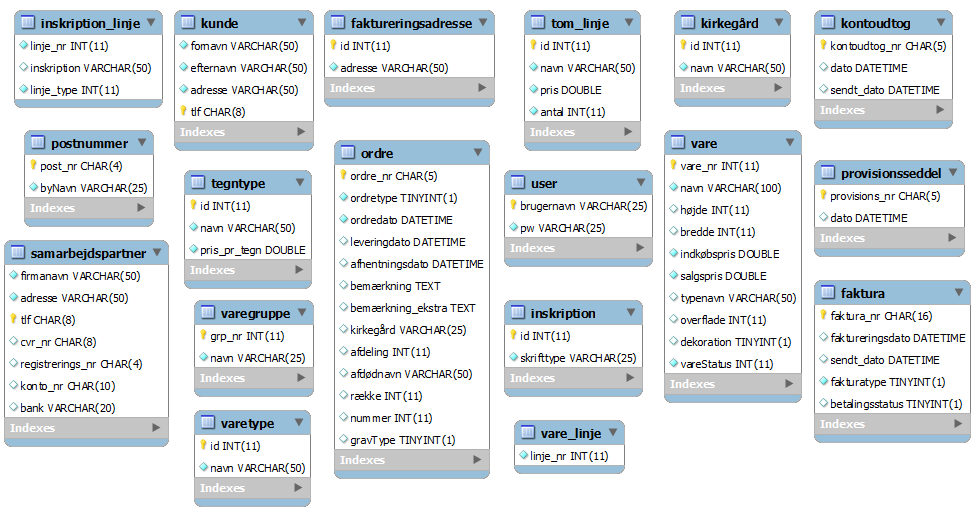
brugernavn varchar(25) not null,

pw varchar(25) not null,

primary key(brugernavn)

) engine = innodb;

### Entitetsdiagram



## Fase 2

### Forbindelser

Mellem varegruppe og vare er der en 1:n forbindelse. En vare hører til netop en varegruppe, mens en varegruppe kan være tilknyttet flere forskellige varer. Der er ligeledes en 1:n forbindelse mellem vare og varetype, idet en vare kan indeholde en varetype og en varetype kan være tilknyttet flere varer.

Mellem vare\_linje og vare er der en 1 : 1 forbindelse. Det samme gælder for vare\_linje og tom\_linje, samt vare\_linje og inskription. En vare\_linje kan kun indeholde en vare, en tom\_linje eller en inskription, om omvendt kan en vare kun være tilknyttet en vare\_linje.

Mellem tegntype og inskription er der en 1 : n forbindelse. En inskription kan kun indeholde en tegntype, mens en tegntype kan være tilknyttet flere inskriptioner.

Mellem inskription og inskription\_linje er der en 1 : n forbindelse. En inskription kan indeholde op til fem inskription\_linjer. En inskription\_linje kan kun være tilkyttet en inskription.

Mellem vare\_linje og ordre er der en 1 : n forbindelse. En vare\_linje kan indgå i en ordre, mens en ordre kan indeholde op til otte vare\_linjer.

Mellem kirkegård og ordre er der en 1 : n forbindelse. En ordre til en kirkegård kan kun leveres til en kirkegård, mens en kirkegård kan være tilknyttet flere forskellige ordre.

Mellem kunde og ordre er der en 1 : n forbindelse. Den samme kunde kan bestille flere ordre, men en bestemt ordre kan kun bestilles af en kunde.

Mellem postnummer og kunde er der en 1 : n forbindelse. Det samme gælder for postnummer og samarbejdspartner samt postnummer og faktureringsadresse.

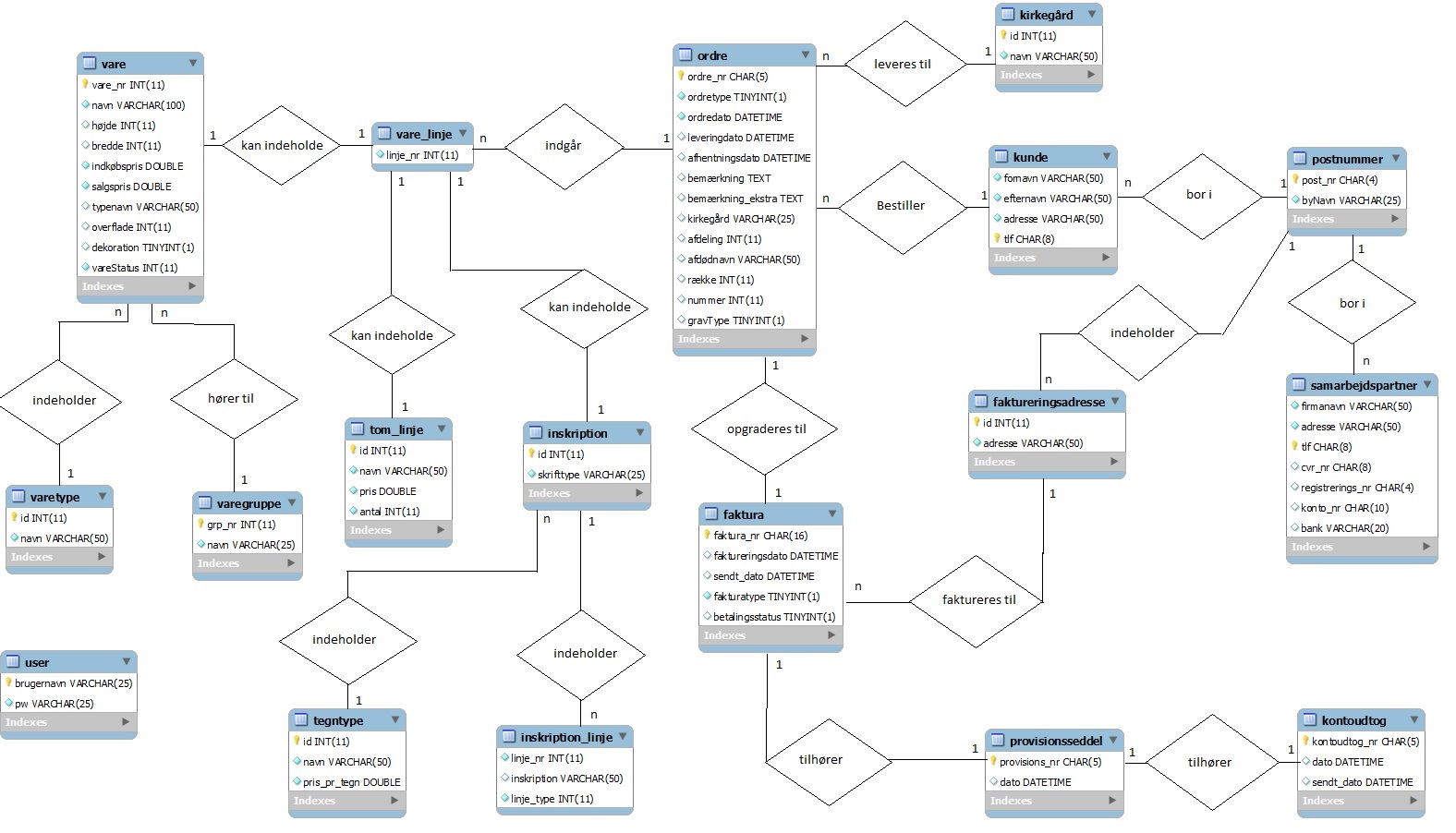
Mellem ordre og faktura er der en 1 : 1 forbindelse. En ordre opgraderes til en faktura og der er derfor ikke mulighed for at en faktura er tilknyttet flere ordre.

Mellem faktureringsadresse og faktura er der en 1 : n forbindelse. En faktura kan kun faktureres til en faktureringsadresse, men en faktureringsadresse kan tilknyttes flere fakturaer.

Mellem faktura og provisionsseddel er der en 1 : 1 forbindelse, idet et kontoudtog kun kan tilknyttes en faktura og omvendt. Det samme gælder for provisionsseddel og kontoudtog.

Angives disse forbindelser i entitetsdiagrammet, får vi følgende.

## ER diagram



## Fase 3

### Beskrivelse af alle 7 punkter i konverteringen

#### Udskrift af SQL-scriptet 2\_konvertering

use bejerholmstenhuggeri;

#fase 1 - Svage entiteter gøres stærke ved at der tilføjes en primær nøgle.

# Har ikke lavet primærnøgler til alle linje tabellerne da dette automatisk kommer

# når vi laver fase 2 i konverteringen

#fase 2 - For hver stærk entitet oprettes en tabel, med registrering af primærnøglen.

#Alle tabeller er oprettet i 1\_entiteter scriptet

#fase 3 - For hver 1:1 forbindelse tilføjes primærnøglen fra den ene tabel som

#fremmednøgle i den anden.

alter table vare\_linje

add column vare\_nr int,

add foreign key(vare\_nr) references vare(vare\_nr),

add column inskription\_id int,

add foreign key(inskription\_id) references inskription(id),

add column tom\_linje\_id int,

add foreign key(tom\_linje\_id) references tom\_linje(id);

alter table faktura

add column ordre\_nr char(5),

add foreign key(ordre\_nr) references ordre(ordre\_nr),

add column provisions\_nr char(5),

add foreign key(provisions\_nr) references provisionsseddel(provisions\_nr),

add column adresse\_id int,

add foreign key(adresse\_id) references faktureringsadresse(id);

alter table faktureringsadresse

add column post\_nr char(4),

add foreign key(post\_nr) references postnummer(post\_nr);

alter table provisionsseddel

add column kontoudtog\_nr char(5),

add foreign key(kontoudtog\_nr) references kontoudtog(kontoudtog\_nr);

#fase 4 - For hver 1:n forbindelse tilføjes primærnøglen fra 1-siden som fremmednøgle

#på n-siden.

alter table vare

add column grp\_nr int,

add foreign key(grp\_nr) references varegruppe(grp\_nr),

add column type\_id int,

add foreign key(type\_id) references varetype(id);

alter table vare\_linje

add column ordre\_nr char(5),

add foreign key(ordre\_nr) references ordre(ordre\_nr),

add primary key(ordre\_nr, linje\_nr);

alter table inskription

add column tegn\_id int,

add foreign key(tegn\_id) references tegntype(id);

alter table inskription\_linje

add column inskription\_id int,

add foreign key(inskription\_id) references inskription(id),

add primary key(inskription\_id, linje\_nr);

alter table kunde

add column post\_nr char(4),

add foreign key(post\_nr) references postnummer(post\_nr);

alter table samarbejdspartner

add column post\_nr char(4),

add foreign key(post\_nr) references postnummer(post\_nr);

alter table ordre

add column tlf char(8),

add foreign key(tlf) references kunde(tlf),

add column kirkegård\_id int,

add foreign key(kirkegård\_id) references kirkegård(id);

alter table faktura

add column bedemand\_tlf char(8),

add foreign key(bedemand\_tlf) references samarbejdspartner(tlf);

#fase 5 - For hver n:m forbindelse oprettes en ny tabel med primærnøglerne fra de to

#tabeller som sammensat primærnøgle og med de to primærnøgler som fremmednøgler.

#Har ingen n:m forbindelser

#fase 6 - For hver flerværdi attribut oprettes en ny tabel med primærnøglen og flerværdi

#attributten som sammensat primærnøgle. Samtidig fjernes flerværdi attributten fra

#den oprindelige tabel.

#Har ingen flerværdi attributter

#fase 7 - Hvis der findes forbindelser hvori der indgår mere end to entiteter skal

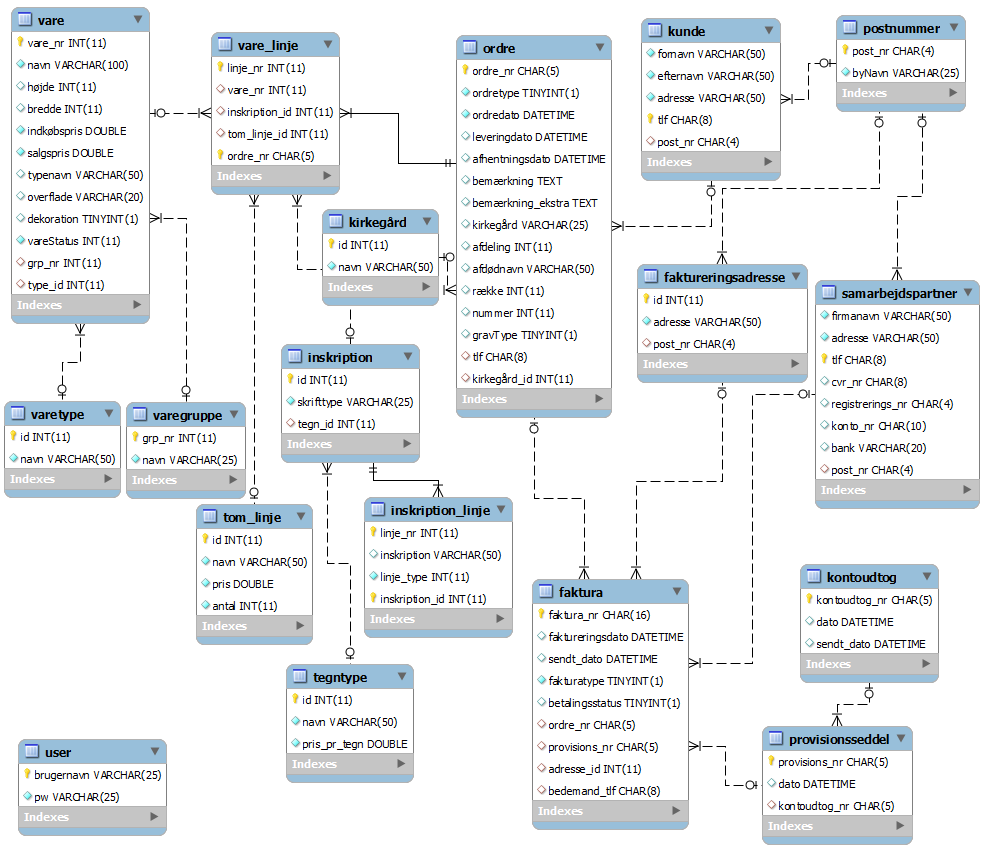
#der oprettes en ny tabel med primærnøglerne fra de deltagende entiteter som

#sammensat primærnøgle.

#Har ingen forbindelser hvor der indgår mere end 2 entiteter

### 

### EER diagram



## Fase 4

### Indsæt data

#### Udskrift af SQL-scriptet 3\_data

use bejerholmstenhuggeri;

insert into postnummer (post\_nr,byNavn)

values

('4200','Slagelse'),

('4300','Holbæk'),

('4700','Næstved'),

('4684','Fensmark');

insert into kunde (post\_nr, fornavn, efternavn, adresse, tlf)

values

('4700','Martin','Hana','Østre Ringvej 40 1. TV','20332836'),

('4700','Thomas','Nielsen','Østre Ringvej 49 2. TV','50111211'),

('4700','Niklas','Nielsen','Gottliebvej 15','31255685'),

('4700','Anette','Stidsing','Eventyrvej 27','28931093');

insert into faktureringsadresse (adresse, post\_nr)

values

('Gottliebsvej 7', '4700'),

('Hejvej 14', '4700');

insert into varegruppe (navn)

values

('Gravsten'),

('Bronce Dekorationer'),

('Granitplader'),

('Fuglebade'),

('Granitvaser');

insert into varetype (navn)

values

('O. Storm Broncedekoration'),

('Strassacker Broncedekoration'),

('Granitplan'),

('Fuglebad'),

('Granitvase');

insert into vare (grp\_nr,navn,højde,bredde,indkøbspris,salgspris,type\_id,overflade,dekoration,vareStatus)

values

('1','Sort Svensk','55','32','950','3300',null,'2',true,'0'),

('1','Mørk Halmstad Hjerte','54','43','850','4600',null,'2',false,'1'),

('1','Lys Halmstad','45','36','650','3200',null,'2',true,'0'),

('1','Paradis','43','38','700','3200',null,'2',false,'0'),

('1','Orion','54','49','800','4700',null,'2',true,'0'),

('1','Orion','43','45','800','4100',null,'2',true,'0'),

('1','Lapland','52','43','850','3400',null,'1',false,'0'),

('1','Lys Multicolor','58','40','650','3200',null,'1',false,'0'),

('1','Lys Halmstad','45','34','650','3200',null,'2',false,'0'),

('1','Halmstad m/ true bånd','53','44','800','5400',null,'1',false,'0'),

('1','Halmstad','53','40','800','3200',null,'2',false,'0'),

('1','Auroa','57','40','900','3500',null,'2',false,'0'),

('1','Halmstad','58','39','8000','3300',null,'1',false,'0'),

('1','Wiscont White','58','42','750','3500',null,'1',false,'0'),

('1','Orion','57','39','800','3300',null,'1',false,'0'),

('1','Natursten m/ true bånd','65','50','350','3400',null,'1',false,'0'),

('1','Lys Multicolor','49','39','650','3200',null,'1',false,'0'),

('1','Sort Svensk','50','38','950','3800',null,'2',false,'0'),

('1','Wiscont White','51','41','750','3400',null,'1',false,'0'),

('1','Blue Pearl','53','34','900','3400',null,'2',false,'0'),

('1','Halmstad','50','38','8000','3200',null,'1',false,'0'),

('1','Natursten','42','33','150','2800',null,'1',false,'0'),

('1','Lys Halmstad','46','36','650','3200',null,'2',false,'0'),

('1','Auroa','58','37','750','3500',null,'2',true,'0'),

('1','Tolga','56','40','600','3300',null,'1',false,'0'),

('1','Halmstad','53','36','470','3200',null,'1',true,'0'),

('1','Halmstad m/true bånd','44','35','470','3200',null,'1',true,'0'),

('1','Rød Multicolor m/ true kant','60','40','650','3400',null,'1',false,'0'),

('1','Orion Urneplade','60','40','850','4500',null,'1',true,'0'),

('1','Kashmir White','68','65','1900','10500',null,'2',false,'0'),

('1','Sort Svensk','70','68','6200','14500',null,'2',true,'0'),

('2','Fugl 181op',null,null,'116','286','1','0',true,'0'),

('2','Fugl 181op',null,null,'116','286','1','0',true,'0'),

('2','Fugl 181op',null,null,'116','286','1','0',true,'0'),

('2','Fugl 181op',null,null,'116','286','1','0',true,'0'),

('2','Fugl 181skrå',null,null,'116','286','1','0',true,'0'),

('2','Fugl 181skrå',null,null,'116','286','1','0',true,'0'),

('2','Fugl n227',null,null,'146','286','1','0',true,'0'),

('2','Blomst 2631v',null,null,'115','500','1','0',true,'0'),

('2','Blomst 373h',null,null,'69','270','1','0',true,'0'),

('2','Blomst 263v',null,null,'189','500','1','0',true,'0'),

('2','Hund 99005',null,null,'175','500','1','0',true,'0'),

('2','Fugl 85328v',null,null,'450','900','2','0',true,'0'),

('2','Fugl 85328h',null,null,'450','900','2','0',true,'0'),

('2','Fugl 85368',null,null,'450','900','2','0',true,'0'),

('2','Blomst 20137+39',null,null,'1100','2200','2','0',true,'0'),

('2','Blomst 31240',null,null,'2150','4300','2','0',true,'0'),

('2','Blomst 29214',null,null,'1200','2400','2','0',true,'0'),

('2','Blomst 31960+20',null,null,'1250','2500','2','0',true,'0'),

('2','Blomst 31960+20',null,null,'1250','2500','2','0',true,'0'),

('2','Blomst 31960+20',null,null,'1250','2500','2','0',true,'0'),

('2','Blomst 31960+28',null,null,'1925','3850','2','0',true,'0'),

('2','Blomst 20611+15',null,null,'425','850','2','0',true,'0'),

('2','Sommerfugl 20367',null,null,'775','1550','2','0',true,'0'),

('2','Hjerte 20408(a)+10',null,null,'550','1100','2','0',true,'0'),

('2','Hjerte 20408(a)+10',null,null,'550','1100','2','0',true,'0'),

('2','Hjerte 20408+10',null,null,'625','1250','2','0',true,'0'),

('2','Hjerte 20408+10',null,null,'625','1250','2','0',true,'0'),

('2','Lampe 40148',null,null,'1475','2950','2','0',true,'0'),

('3','Mørk Halmstad','134','83','1800','0','3','0',false,'0'),

('3','Wiscont White','100','97','1600','0','3','0',false,'0'),

('4','Blå Rønne','22','18','75','350','4','0',false,'0'),

('4','Vonga','25','21','75','350','4','0',false,'0'),

('4','Halmstad','22','19','75','350','4','0',false,'0'),

('4','Rød Bohus Hjerte','27','26','75','450','4','0',false,'0'),

('5','Rund Sort Svensk','0','0','360','650','5','0',false,'0'),

('5','Rund Sort Svensk','0','0','360','650','5','0',false,'0'),

('5','Rund Sort Svensk','0','0','360','650','5','0',false,'0'),

('5','Rund Sort Svensk','0','0','360','650','5','0',false,'0'),

('5','Rund Sort Svensk','0','0','360','650','5','0',false,'0'),

('5','Rød Bohus','0','0','347','650','5','0',false,'0'),

('5','Vonga','0','0','347','650','5','0',false,'0');

insert into kirkegård(navn)

values

('Næstved Kirkegård'),

('Slagelse Kirkegård'),

('Korsør Kirkegård'),

('Paris Kirkegård');

insert into ordre (tlf, ordre\_nr,ordretype,ordredato,leveringdato,afhentningsdato,bemærkning,bemærkning\_ekstra,kirkegård\_id,afdeling,afdødnavn,række,nummer,gravType)

values

('20332836','00001',true,'2013-04-30 12:50:32','2013-05-10 12:50:32','2013-05-15 12:50:32','Bemærkning','Bemærkning\_ekstra','1','1','Gunner','10','1',true),

('50111211','00002',true,'2013-04-1 13:00:00','2013-05-11 13:00:00','2013-05-16 13:00:00','Bemærkning','Bemærkning\_ekstra','2','1','Gunner','10','2',true),

('31255685','00003',true,'2013-04-2 13:00:00','2013-06-11 13:00:00','2013-05-17 13:00:00','Bemærkning','Bemærkning\_ekstra','3','1','Gunner','11','1',true),

('31255685','00007',true,'2013-04-2 13:00:00','2013-06-11 13:00:00','2013-05-17 13:00:00','Bemærkning','Bemærkning\_ekstra','3','1','Gunner','11','1',true),

('28931093','00004',true,'2013-04-2 13:00:00','2013-06-11 13:00:00','2013-05-18 13:00:00','Bemærkning','Bemærkning\_ekstra','4','1','Gunner','11','2',true);

insert into tegntype (navn,pris\_pr\_tegn)

values

('Indh. m/ alm. farve','65.00'),

('Indh. m/ guld','90.00'),

('Indh. m/ guld i ru sten','115.00'),

('Faks. indh. m/ farve','98.00'),

('Faks. indh. m/ guld','118.00'),

('Indh. i egen sten- m/ alm. farve','75.00'),

('Indh. i egen sten- m/ alm. guld','118.00'),

('Håndtegnet & Håndhugget','130.00'),

('Håndtegnet & Sandblæste','90.00'),

('Bronce bogstaver','75.00'),

('Opmaling af eksisterende','35.00'),

('Oppudsning af Bronce','55.00'),

('Opforgyldning i pol. sten','65.00'),

('Opforgyldning i ru sten','95.00');

insert into inskription(tegn\_id, skrifttype)

values

('2','ARIAL'),

('4','OLD ENGLISH'),

('7','Agmena'),

('10','Flexo');

insert into inskription\_linje(linje\_nr, inskription\_id, linje\_type, inskription)

values

('1','1','1','Hej med dig'),('2','1','1','min mor'),('3','1','1','og du er'),('4','1','1','den bedste hej'),('5','1','1','med dig far'),

('1','2','2','Anette Mi Stidsing'),('2','2','2','\* 25. 6. 1992 + 1. 1. 2013'),('3','2','1','Niklas Renner Nielsen'),('4','2','1','\* 25. 6. 1992 + 1. 1. 2013'),('5','2','2','Hvil i fred'),

('1','3','1','Hej'),('2','3','1','mor'),('3','3','3',''),('4','3','3',''),('5','3','3',''),

('1','4','1','Kagemand 2000-2001'),('2','4','1','hvil i fred'),('3','4','0',''),('4','4','0',''),('5','4','0','');

insert into tom\_linje (navn,pris,antal)

values

('Navn1','1000','1'),

('Navn2','2000','2'),

('Navn3','3000','1'),

('Navn4','4000','1');

insert into vare\_linje (linje\_nr, vare\_nr, inskription\_id, tom\_linje\_id, ordre\_nr)

values

('1','2',null,null,'00001'),

('2',null,null,'1','00001'),

('3',null,'2',null,'00001'),

('1',null,'1',null,'00004');

insert into samarbejdspartner (post\_nr, firmanavn,adresse,tlf,cvr\_nr,registrerings\_nr,konto\_nr,bank)

values

('4700','Bedemand ApS','Bedemandsvej 1 4220 Korsør','58350001','12943790','2190','0754567300','Nordea'),

('4700','Bedemand Berner','Fiktivvej 666','11112226','16276482','1232','6070123457','BankenB'),

('4700','Bedemand Berner','Fiktivvej 666','11112222','16276483','1233','6070123458','BankenC'),

('4700','Bedemand Berner','Fiktivvej 666','11112224','16276484','1234','6070123459','BankenD');

insert into kontoudtog (kontoudtog\_nr,dato,sendt\_dato)

values

('00002','2013-04-30 12:50:32','2013-04-30 12:20:32'),

('00003','2013-04-30 13:50:32','2013-04-30 13:20:32'),

('00004','2013-04-30 14:50:32','2013-04-30 14:20:32');

insert into provisionsseddel (provisions\_nr, kontoudtog\_nr, dato)

values

('00002','00002','2013-05-02 15:20:32'),

('00003','00003','2013-05-03 16:20:32'),

('00004','00004','2013-05-04 17:20:32');

insert into faktura (bedemand\_tlf, provisions\_nr, ordre\_nr, faktura\_nr,faktureringsdato,sendt\_dato,adresse\_id,fakturatype,betalingsstatus)

values

(null, null,'00002', '0050111211-00002','2013-04-30 12:50:32','2013-05-01 12:50:32',null,false,false),

(null, null,'00003', '0031255685-00003','2013-04-30 13:50:32','2013-05-01 13:50:32',null,false,true),

('58350001', '00004','00004', '0028931093-00004','2013-04-30 14:50:32','2013-05-01 14:50:32',null,true,false);

insert into user (brugernavn,pw)

values

('martinhana','martin10'),

('Mahias BejerHolm','Mathias20');

## 

## Fase 5

### Test

#### Udskrift af SQL-scriptet 4\_test

use bejerholmstenhuggeri;

-- Test der kontrollerer at primærnøgler samt fremmednøgler korrekt registrerede.

update ordre

set ordre\_nr = 00009

where ordre\_nr = 00001;

update ordre

set kirkegård\_id = 7

where kirkegår\_id = 3;

update kunde

set tlf = 29038940

where tlf = 28931093;

update kunde

set post\_nr = 5695

where post\_nr = 4700;

update vare

set vare\_nr = 50

where vare\_nr = 20;

update vare

set grp\_nr = 9

where grp\_nr = 3;

update faktura

set ordre\_nr = 7

where ordre\_nr = 1;

update samarbejdspartner

set tlf = 28282828

where tlf = 58350001;

-- select

select navn, sum(indkøbspris), min(indkøbspris), max(indkøbspris)

from vare;

select vare\_nr, navn, sum(salgspris), max(salgspris), min(salgspris)

from vare

group by vare\_nr

having max(salgspris) < 850;

select \* from kunde where post\_nr = 4700;

Select \* from kunde

where efternavn like '%sen%';

-- delete

delete from ordre

where ordre\_nr = '00007';

delete from vare

where vare\_nr = 8;

-- insert

insert into postnummer(post\_nr, byNavn)

values

('8981', 'Spentrup'),

('9000', 'Ålborg');

-- update

update kunde

set adresse = 'Christiansvej 43'

where adresse = 'Eventyrvej 27';

-- queries der udtrækker data fra mere end en tabel.

select ordre.ordre\_nr, ordre.leveringdato, kunde.tlf, kunde.fornavn, kunde.efternavn

from ordre, kunde

where ordre.tlf = kunde.tlf

group by ordre.ordre\_nr;

select \*

from (varegruppe left outer join vare on varegruppe.grp\_nr = vare.grp\_nr);

select varegruppe.navn, vare.navn, vare.vare\_nr

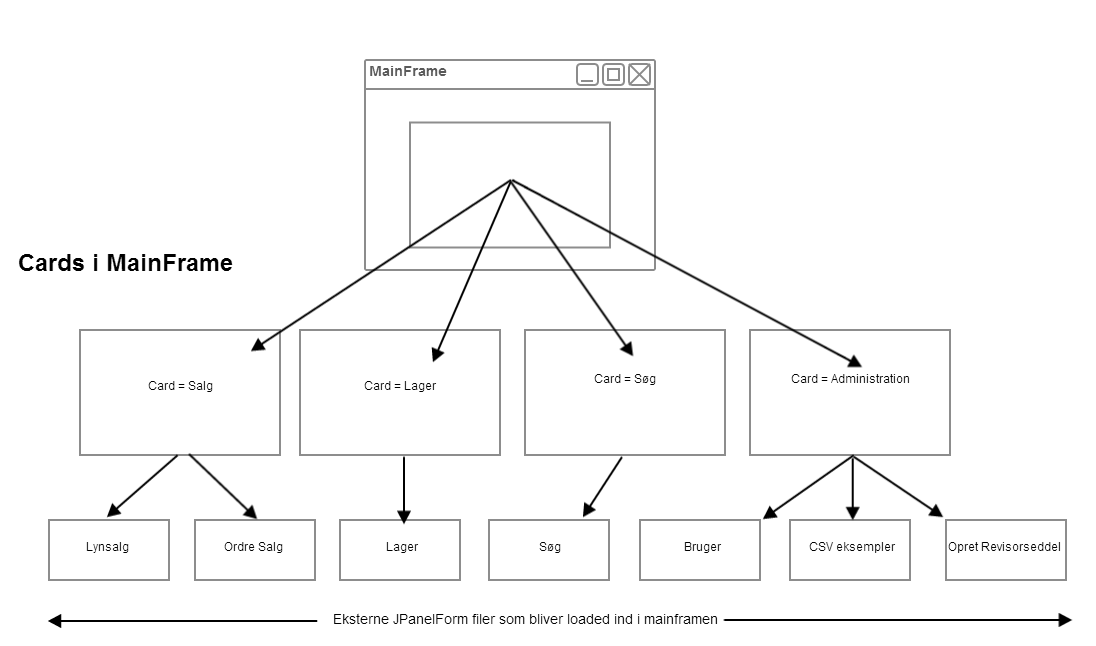
from (varegruppe join vare on varegruppe.grp\_nr=vare.grp\_nr)

where varegruppe.grp\_nr = 1;

# Programmering

## Beskrivelse af program

Ved projekt start valgte vi, at programmet skulle laves i cardlayout, hvor alle card blev oprettet direkte i gui filen. På den måde kunne vi bare skifte card alt efter, hvor man skulle hen i programmet. Med denne løsning løb vi hurtigt ind i problemer. Et af de første problemer var, at vi havde alt i en fil, det var derfor kun muligt for en person at kunne redigere i dokumentet af gangen. Denne problemstilling gjorde, at det ikke ville være muligt at færdiggøre projektet inden for afsatte tidsramme. Ydermere, som der blev tilføjet kode til filen opdagede vi omfanget af hvor mange linjer kode, der ville komme til at være i selve filen. Det ville ikke muligt, at opretholde et overblik i flere tusinde linjer kode i filen, og samtidig ville programfejl være kritiske at lokalisere.  
Vi kom derfor til enighed om at dele programmet ud i mindre dele således, at vi alle kunne få et ansvarsområde hver og arbejde på den del uafhængigt af hvad de andre arbejder med. Det er fremvist i figuren nedenfor.

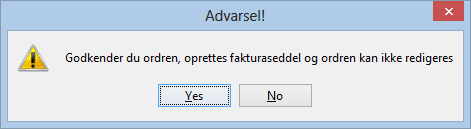


Alle de eksterne JPanelForms bliver tilføjet til mainframen og lavet om til cards, på den måde kan vi nemt skifte rundt imellem dem når der navigeres rundt i menuen. Denne løsning gør også arbejdet med fejlfinding nemmere, da man kan henvise til den enkelte fil og derfra fejlfinde i en langt mindre mængde kode.

Vi har valgt i større omfang at inkorporere dialog bokse i vores system. Til grund for denne beslutning ligger, at vi gerne vil være sikre på, at alle større hændelser i systemet som brugeren foretager sig skal bekræftes. På den måde sikrer vi os at brugeren altid bliver gjord opmærksom på både, at det er den rigtige handling han udfører og hvad det medfører, men samtidig minimerer det også fejlhandlinger programmet som fx. kunne forekomme ved misklik.

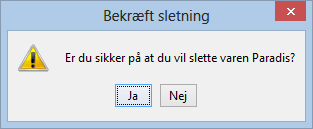
Eksempler på dialog bokse i systemet:

Lynsalg:  
Når brugeren er kommet til ordrebekræftelsen, og der trykkes godkend, bliver denne besked vist til brugeren:



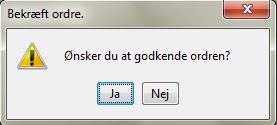
### Lager:

Når en vare er valgt i listen og man trykker på slet knappen vises denne besked til brugeren:



### Ordresalg:

Når brugeren trykker godkend i ordrebekræftelsen, vil denne besked vises:



Komplet liste over dialog bokse i programmet.– OPDATER NÅR PROGRAM ER FÆRDIGT

* Mainframe
  + Tilbage til hovedmenu
* Lynsalg
  + Godkend Ordre
  + Annuller Ordre
  + Manglende valgt vare
* Ordre Salg
  + Godkend Ordre
  + Annuller Ordre
* Lager
  + Bekræft opret vare
  + Annuller opret vare
  + Slet vare
  + Bekræft rediger vare
  + Annuller rediger vare
* Søg
  + Ændre betalingsstatus på igangværende faktura
  + Slet Faktura
  + Slet Ordre
  + Annuller Redigering af ordre
  + Godkend redigering af ordre
  + Annuller opgradering af faktura
  + Godkend opgradering af faktura

## Klasser og centrale metoder

## Login

Det vindue der åbnes når programmet starter. Den tjekker om den kan connect til databasen, hvis den ikke kan skifter den card til DB card, hvor man selv kan indsætte de korrekte oplysninger. Hvis den kan connect til databasen viser den Login card for brugeren.

***jButton\_LoginActionPerformed***Ved klik på login knappen tjekker den i databasen om brugernavnet findes, hvis det gør tjekker den om det indtastede password stemmer overens med det der er angivet i databasen. Er det dét korrekte password oprettes der et Mainframe objekt og der kaldes en dispose på login.

# Mainframe

Mainframe bliver åbnet af login og bliver kaldt med et User objekt samt et DBhandler objekt. User objektet bliver brugt til at holde styr på hvem der er logged ind. Mainframen håndterer navigationen rundt i programmet som beskrevet tidligere i hvordan vi har valgt at løse opgaven.

Centrale metoder

***jButton\_LagerOpgørelseActionPerformed();***  
Ligger i Administrations cardet og bruges til at lave en samlet lageropgørelse af varegrupperne. Metoden kalder OpretRevisorSeddel med datoen for opgørelse, året samt dbhandler. Den kører herefter metoden i OpretRevisorSeddel: genererRevisorSeddel som oprettet pdf filen. Ved korrekt oprettelse åbnes pdf filen.

***jButton\_LogUdActionPerformed()***Oprettet et nyt Login object og sætter det til visible= true samtidig kaldes der en dispose() på framen så den lukkes.

***vishovedmenu()***Bruges når der klikkes på Hovedmenu knappen eller hvis der skal skiftes efter en handling. Den kalder også metoden opretPaneler som resetter alle Paneler ved at sætte dem = new. Det er en public metode så den kan kaldes i Panelerne.

Der henvises til UML diagrammet, under systemudvikling i afsnittet designdokument.

DBConnection klassen

Klassen DBConnection har ansvar for at håndtere forbindelsen til databasen. På den måde kan vi ved programmets start oprette et database- objekt, og dermed oprette forbindelse til databasen. Det betyder at hver gang vi skal hente eller indsætte noget i databasen, så kan vi bruge objektets metoder.

#### connect()

Metoden opretter forbindelse til de parametre som er medtaget fra constructoren (String db\_host, String db\_port, String db\_name, String db\_user, String db\_pw) og returnerer true eller false.

getData(String getdata)  
Medtoden bruger vi når vi skal hente data fra databasen. Man kalder funktionen med en sql-streng som så returnerer resultatet i et ResultSet.

#### setData(String getdata)

Metoden bruger vi når vi skal indsætte data i databasen. Man kalder funktionen ved at give den en sql-streng, der så skal bruges til indsættelsen.

DatabaseObjectHandler klassen

Klassen DatabaseObjectHandler håndterer de SQL-queries der laves i programmet. Den har forskellige metoder som kan hente modelobjekter ud af databasen, indsætte dem i databasen, redigere og slette dem.

Klassen er en overbygning af DBConnection klassen. Vi har lavet klassen så vi har alle metoder til at hente de førnævnte objekter ud af databasen i modsætning til at lave dem i hvert enkelt klasse hvor vi skal hive dem ud. Vi har valgt at dele SQL-queries op i lag, altså getKunde kalder getPostnummer, i stedet for at lave udtræk af flere tabeller. Det kunne vi gøre anderledes næste gang vi laver et projekt da de fleste get Metoder så kun ville kræve en SQL-streng/metode i modsætning til flere lag.

public ArrayList<Vare> getFiltreretVareListe(int grp\_nr, int minHøjde, int maxHøjde, int minBredde, int maxBredde, double minPris, double maxPris, int vareStatus)  
Metoden henter en ArrayList af Vare objekter ud af databasen, som er filtreret ud fra de parametre som constructoren er kaldt med. Der opbygges en SQL streng ud fra parametrerne ved at den tjekker om de forskelling parametre er 0, og hvis de ikke er lægges det paramter til SQL-strengen. De metoder der er beskrevet i denne klasse er udvalgt fordi de beskriver de forskellige slags metoder vi har, da vi har mange gengangere til de forskellige objekter.

public String createOrdre(Ordre ordre)Metoden bruges til at oprette en ordre i databasen ud fra det Ordre objekt som constructoren bliver kaldt med. Metoden starter med at kalde createKunde metoden med Ordre objektets Kunde objekt som derefter vil oprettet en kunde hvis den ikke findes, eller redigere den hvis den findes i forvejen. Derefter tjekker metoden om en en ordre med det valgte ordre nummer findes i forvejen, hvis det gør det smider den en fejl og stopper metode, hvis den ikke gør går den videre. Derefter tjekker metoden om der er et Kirkegård objekt i Ordre objektet, hvis der opretter den ordren i databasen med et kirkegård id, hvis der ikke er opretter den det uden. Til sidst tager metoden Vare\_Linje ArrayList’en fra Ordre objektet og opretter dem enkeltvist med Ordre objektets ordre\_nr og et linje nr, samt detaljerne om varelinjerne.

#### public int getMaxVareNr()

Metoden bruger vi til at finde det højeste varenummer der findes i databasen, som den så returnerer som en integer. Vi bruger den til at finde den sidste vare vi har oprettet så vi kan knytte den på andre objekter, i stedet for at søge på navn el.lign.

#### public void updateVareStatus(Vare vare)

Metoden bruger vi til at opdatere varestatus på en vare når vi opretter et ordre objekt, så databasen registrerer at varen ikke er på lager mere, eller omvendt hvis det er en annullering af en ordre. Vi bruger en update som finder varenummeret og sætter varestatus for den vare til den nye varestatus som er valgt.

#### public String getNextOrdreNr()

Den finder det det højeste ordre nummer i databasen og laver det om til en integer så vi kan lægge 1 til hvorefter vi sætter nuller på igen for at gøre det til det næste ordre nummer. Vi valgte at gøre ordrenummer til en String fordi at vi skulle have de ekstra nuller på, men efterfølgende har vi kunne se at det ikke var nødvendigt da vi bare kunne have sat nullerne på i GUI/PDF’er når vi skulle vise dem.

#### public void deleteOrdre(Ordre ordre)

Metoden bruger vi til at slette ordre i databasen. Vi bruger en delete til at slette alle entries der passer til det ordre nummer der er i ordre objektet.

### LynSalg

Lynsalg bruges til at sælge varer direkte fra lageret uden nogle form for arbejde skal udføres for dem.  
Varene vælges ud fra jComboBox\_Lynsalgvaregruppe som indeholder varegrupperne og har et event på actionperformed tilknyttet. Hver gang den ændres tilføjer den varene fra den valgte varegruppe ind i jComboBoxLynsalgVare.  
  
Panel\_LynSalgLinje  
Panel\_LynSalgLinje er tilknyttet Lynsalg og bruges til at vise den valgte vare. Klassen indeholder to jlabels en som sættes til varenavn og en der sætter pris samt tre knapper.



* Slet: Kalder *removepanel()* i LynSalg klasses, hvilket er beskrevet under centrale metoder.
* Ryk op: Kalder flytvarelinje() i LynSalg, hvilket er beskrevet under centrale metoder.
* Ryk ned: Kalder flytvarelinje() i LynSalg, hvilket er beskrevet under centrale metoder.

***jButtonLynsalgTilføjActionPerformed***  
Ved klik på knappen jButtonLynsalgTilføj tilføjes den valgte vare fra jComboBoxLynsalgVare til Arraylisten valgteVare\_lynsalg. Samtidig oprettes der et objekt af Panel\_LynSalgLinje som kaldes med den valgte vare dette objekt puttes ind i ArrayListen panel. Hvis valgteVare\_lynsalg Arrayet.size > 8 sættes knappen til disabled og sættes først til enabled når der bliver fjernet en vare fra arrayet. Varen bliver også fjernet fra Comboboksen så det ikke er muligt at vælge den samme vare to gange.

***Drawpanel(JPanel panel)***Bruges til at tilføje alle de objekter af Panel\_LynSalgLinje, der er blevet tilføjet til ArrayListen. De bliver tilføjet til enten jPanel\_VareLinjer eller jPanel\_OversigtVarer alt efter hvilket panel den bliver kaldt med.

***removepanel(Panel\_LynSalgLinje panel)***Kaldes igennem Panel\_LynsalgLinje, hvor der er en knap til at fjerne en valgt vare fra listen. Ved klik på knappen fjerner den varen fra ArrayListen ValgteVare\_lynsalg samt fra det panel som varen ligger i. Samtidig sættes jComboBox\_Lynsalgvaregruppe’s selected index til til det som den allerede står på. På den måde opdateres varerne, og den vare der er fjernet nu kan vælges igen i comboboksen.

***jButton\_godkendActionPerformed***Ved klik på knappen jButton\_godkend tjekkes der om der er nogle varer i ArrayListen valgteVare\_lynsalg, hvis den er tom kommer der et Joptionpane og informerer om det ikke er muligt at oprette en ordre uden nogle varer. Ved varer i arrayet oprettes et ordre objekt med butikken som kunde samt objekter af Varelinjer af alle de valgte varer. Der bruges constructoren i ordre til lynsalg Ordre(kunde, varelinjer) til at oprette objektet. Efter ordre objektet er oprettet korrekt kaldet metoden createOrdre() som opretter ordren i databasen. Efterfølgende kaldes createFaktura() med den nyoprettede ordre. Hvis alt oprettes uden fejl, bliver en pdf med fakturaen lavet og åbnet.

***flytvarelinje(Panel\_LynSalgLinje i, boolean flytvej)***Kaldes fra Panel\_LynSalgLinje og bruges til at rykke en vare op eller ned i ArrayListen valgteVare\_lynsalg samt panel rækkefølgen i ArrayListen jPanel\_OversigtVarer. Flytvej bestemmer om det er op eller ned. Collection.swap bruges til at bytte rundt på to pladser i Arraylistene. Efterfølgende kaldes drawpanel så ændringen bliver opdateret i jpanelerne der viser varelinjerne.

### Panel\_OrdreSalg og Panel\_OrdreSalgLinje

Klassen Panel\_OrdreSalg har ansvar for at gøre det muligt for brugeren, at oprette en ordre som gemmes i databasen. Den har derudover til opgave at oprette en PDF fil for ordren og gemme denne. Derudover indeholder klassen funktioner som gør det muligt at oprette en ny kunde, idet der indtastet oplysninger eller at redigere i en eksisterende kunde.

Panel\_OrdreSalgLinje er oprettet så det var muligt at tilføje et vare\_linjeobjekt som et panel med navn, pris samt en knap til at slette vare\_linjen igen. Et Panel\_Ordresalg objekt bliver kaldt med et vare\_linje objekt samt et panel og benyttes i Panel\_Ordresalg, således at en valgt vare vil blive vist som et Panel\_OrdreSalgLinje i et panel i klassen. Denne klasse har ingen metoder.

#### private void hentLister() - private void fyldVaregruppe()

Metoderne har til formål at gøre det muligt at hente varegrupper ud fra databasen og tilføje dem til comboboxe.

hentlister() forsøger at hente alle varegrupper ud fra databasen og lægge dem ind i ArrayListvaregruppelisten, ved at bruge en databaseobjecthandler: dbhandler.getvaregruppe(). Derefter løber fyldVareGruppe() listen igennem i en forløkke og add'er dem ind i jComboxene.

#### public void drawpanel(JPanel jpanel)

Metoden benyttes og i Panel\_lynsalg, hvor den er beskrevet.

#### public void removepanel(Panel\_OrdreSalgLinje i)

Metoden benyttes og i Panel\_lynsalg, hvor den er beskrevet.

#### removepanel(Panel\_OrdreSalgLinje i)

Metoden benyttes og i Panel\_lynsalg, hvor den er beskrevet.

#### private void jButton\_videre\_ordresalgActionPerformed(java.awt.event.ActionEvent evt)

Her benyttes metoden skriftpanel()

#### private void skiftpanel()

Metoden tjekker først om der er indtastet kundeoplysninger og derefter om der er valgt enten er en tilføjelse eller en ny sten, således at ordretypen kan bestemmes.

Metoden benytter flere små metoder:

##### private Vare\_linje getVærkstedstimerVarelinje()

##### private Vare\_linje getTransportVarelinje()

##### private Vare\_linje getRensningVarelinje()

##### private Vare\_linje getLeveringVarelinje()

##### private Vare\_linje getAfhentningVarelinje()

##### private Vare\_linje getInskriptionVarelinje()

##### private void opretTilføjelselinje()

##### private void opretInskription\_linje()

Get- metoderne sørger for at oprette de forskellige objekter (tom\_linje/inskription), og derefter et vare\_linje objekt som de returnerer. getVærkstedstimerVarelinje() opretter f.eks. først et tom\_linje objekt, og derefter et vare\_linje objekt som indeholder dette. getInskriptionVarelinje() opretter først en ArrayList< Inskription\_linje> inskription\_linjeListe, idet en inskription kan indeholde flere inskription\_linje objekter. Den tjekker derfor først om brugeren han indtastet noget i textfelterne til inskription\_linjer, og opretter dem som objekter, hvorefter den tilføjer dem inskription\_linjeListe. Til sidst opretter den et inskription objekt, og derefter et vare\_linje objekt med inskriptionen i.

Metoderne opretTilføjelselinje() og opretInskription\_linje() tjekker først hvilke tilføjelser brugeren har valgt. Ud fra hvad der er indtastet, benytter den, den nødvendige get-metode for at få et vare\_linje objekt. Vare\_linje objektet tilføjes valgte\_vareListen, tilføjer denne panelListen og kalder metoden drawpanel() således at et panel\_ordreSalgLinje komponent bliver tilføjet panelet med oversigt over valgte vare. Til sidst tilføjer den vare\_linjen til valgtevareListen.

#### private void opdaterVareListe() og private void opdaterVareListeNySten()

Metoderne benyttes i events, som er tilknyttet jComboxene med varegrupper. Da der er to forskellige jPanels (jPanel\_ordre og jPanel\_ordre\_linje) blev der derfor lavet to metoder, idet metoden har til formål at opdatere samt hente varelister ud fra en en valgt varegruppe i to combobox. Metoderne kunne være lavet anderledes, idet den eneste forskel er, at det er to forskellige comboboxe som benyttes. Man kunne derfor lavet dem som en samlet metode som kaldes med en jCombobox i stedet.

opdaterVareListe() benyttes også i en event tilknyttet jPanel\_valgteVare\_ordresalg, idet et component fjernes.

Metoderne forsøger at hente en vareliste ud ved at bruge dbhandler.getVareListe, som returnere en ArrayList af vare, hvor varene er tilknyttet varegruppen. Varegruppen sættes til den varegruppe som er valgt i comboboxen med varegruppelisten.

Den løber varelisten igennem i en forløkke. Inde i den løber den ArrayListen valgte\_vare igennem. Hvis valgte\_varelisten ikke er tom, tjekker den om objekter i de to lister har det samme varenummer. Har de det, fjernes de fra vareListen.

Til sidst fylder den varelisten op, men alle vare med varestatus 0.

Hvis tilføjelse er valgt oprettes der en tom\_linje vare\_linje med de valgte

#### jButton\_tilføj\_vare\_ordreSalgActionPerformed(java.awt.event.ActionEvent evt)

Der tjekkes først om checkbox\_redigerPris er valgt. Er den valgt, henter den varen ud fra jComboBoxen i et Vare objekt. Derefter laver den en vare.setSalgspris() ud fra det indtastede tal i jTextfield\_redigerPris. Den forsøger så at redigere varen i databasen således dbhandler.editVare(vare).

Den tilføjer derefter den valgte vare til valgtevare\_listen, og opretter et panel\_ordreSalgLinje, som sættes ind i panelet med valgte vare.

#### jButton\_godkend\_ordresalgActionPerformed(java.awt.event.ActionEvent evt)

Der oprettes først et postnummer objekt, et kunde objekt og et ordre objekt som kalder med kunden og valgtevarelisten.

Metoden tjekker hvilken ordretype der er valgt. Er ny sten valgt, sættes ordretypen til true. Er tilføjelse valgt sættes den til false. Derefter tjekker metoden om gravsten er valgt. Er den valgt vil den indsættes kirkegårdsinformationer i ordren. Metoden tjekker så om der er udfyldt tekst i bemærkning felterne, og tilføjer dette ordre hvis de er.

Den opretter derefter et timestamp ved hjælp af utility klassen, til både leverings- samt afhentningsdato alt efter hvilken dato der er valgt i comboxe.

Når de nødvendige objekter er oprettede til oprettese af en ordre og de valgte variable er sat, forsøger metoden at lave en dbhandler.createordre(ordre), således at ordren gemmes i databasen. Den henter derefter ordren ud igen, for at oprette en ordrePDF. Dette gør den ved hjælp af metoden ordrePDF.genererOrdre("ordrePDF.pdf"). Derefter oprettes en fil, som gemmes, og derefter åbnes PDF'en.

Til sidst går den til start menuen ved at bruge frame.vishovedmenu().

Panel\_Lager klassen

Klassen Panel\_Lager har ansvar for lager GUI’en som skal kunne vise alle varer på lager, filtrere dem, og slette/redigere/oprette varer, samt sende brugeren videre til LynSalg hvis de vil købe en vare.

#### public final void getFiltreretVareListe()

Denne metode filtrerer den vareliste vi har til højre ved hjælp af de parametre vi indtaster som filtre nede i varefilter panelet. Den bruger metoden getFilteretVareListe() fra objecthandleren.

#### public boolean visAcceptDialog(String message, String title)

Denne metode opretter en dialog boks som man kan klikke Ja eller Nej til. Metoden kaldes med en message-String som er den besked vi vil have brugeren til at tage stilling til samt en title-String som bruges til at vise titlen på dialogboksen. Når brugeren har trykket ja eller nej returnerer metoden en boolean som vi så kan bruge til at vælge hvad koden skal gøre efter det.

#### public final void opdaterVareLinjePanel()

Metoden fjerner alle LagerVare paneler som vi har til venstre i gui’et hvor den derefter opdaterer den med de Vare objekter som findes i den filtrerede vareliste.

Validering og exceptionhandling vil blive beskrevet i det passende afsnit.

Panel\_LagerLinje klassen

Klassen Panel\_LagerLinje er det panel vi bruger til at vise de enkelte varer i varelinje panelet i lager panelet. Det kaldes med en vare som den så udskriver på panelet, samt et Panel\_Lager som den så kan bruge til at køre dets metoder med.

#### private void varePanelMouseReleased(java.awt.event.MouseEvent evt) Metoden kalder Panel\_Lager’s enterInfo(Vare vare) metode som indtaster vare objektets info til højre I Panel\_Lager hvor vi har vores vare info panel.

JDialog\_LagerVare klassen

Klassen bruges til at redigere/oprette varer til Lageret. Det åbnes når man trykker på Slet vare/Rediger vare i Panel\_Lager’s vareinfo panel. Hvis klassen bliver kaldt med en Vare til den redigere denne vare og indtaste dens info i JDialogen’s TextField’s. Hvis klassen ikke bliver kaldt med en vare er det en oprettelse af en vare og GUI’ens TextField’s vil derfor være tomme

private void opretVareButtonActionPerformed(java.awt.event.ActionEvent evt)  
Metoden kalder en validerer først input fra TextFields og hvis infoen er valid vil den poppe op med en visAcceptDialog() som man kan annullere, hvorefter der ikke sker noget, eller acceptere, hvorefter metoden vil prøve at oprette en vare hvis det er en opret vare dialog eller redigere en vare hvis det er en rediger vare dialog.

# Søg

Klassen søg har til formål at give brugeren en oversigt over igangværende ordre samt fakturaer. Yderligere er det muligt for bruger at slå en ordre eller faktura op udfra deres nr samt se alle ordre der hører til en kunde.  
Søg klassen er delt op i flere klasser som alle bliver kaldt fra Panel\_Søg, disse er beskrevet nedenfor.

**Panel\_Søg\_Ingangværende – JpanelForm**Der bliver oprettet objekter af denne klasse når der trykkes på en af de to radio buttons i søg. Klassen har to constructors. Den ene kaldes med et ordre objekt, den anden med et faktura objekt. Alt efter hvilken constructor objektet oprettes med, kommer de relevante oplysninger i panelet om ordren/fakturaen, samt en knap til at åbne dem i et Jdialog vindue. Dette er beskrevet under Jdialog senere i dette afsnit. Objekterne bliver efter oprettelse tilføjet til et jpanel i Søg, som viser en liste over igangværende fakturaer/ordrer.

**Panel\_Søg\_Resultat - JpanelForm**Denne klasse bruges til at oprette de objekter som bliver fundet i forbindelse med, at der trykkes på søge knappen i søg. Alt efter hvad der er valgt i jComboBox\_VælgType kalder den en af de to constructors med entet et ordre eller faktura objekt. Relevante oplysninger om faktura/ordre udskrives i panelet sammen med en vis knap der åbner entet JDialog\_SøgFaktura eller JDilog\_SøgOrdre, som bliver beskrevet efterfølgende. Det oprettede objekt bliver tilføjet til jPanel\_SøgeResultat i søg.

**JDialog\_SøgFaktura**Denne Jdialog klasse bruges til at vise oplysninger omkring en faktura som bliver sendt med når objektet af klassen oprettes. Det er samtidig også i denne klasse at alt administration af fakturaen forgår.

De centrale metoder i klassen:

***jButton\_BetaltActionPerformed*()**  
Bruges til at ændre betalingsstatus på fakturaen når der klikkes på knappen jButton\_Betalt.  
Hvis fakturaen ikke er betalt sætter den betalingsstatus = true, hvis fakturaen er betalt sætter den betalingsstatus = false på faktura objektet og kalder metoden i Databaseobjektandler editfaktura(Faktura faktura) som opdaterer oplysningerne i databasen.

***jButton\_SletActionPerformed()***Bruges til at slette en faktura. Ved klik på knappen jButton\_Slet kaldes metoden i Databaseobjekthandler deletefaktura(Faktura faktura), som sletter fakturaen samt provisionsseddel og kontoudtog fra databasen hvis de findes.

Der er samtidig tre knapper som ved klik generer henholdsvis faktura, provisions – og kontoudtogsseddel i pdf format. Faktura og provisions knapperne er kun enabled såfremt de er tilknyttet fakturaen.

**JDialog\_SøgOrdre**Denne Jdialog klasse bliver ligesom SøgFaktura brugt til at vise en ordre samt de mulige handlinger der hører under denne.

De centrale metoder:

**startOpgradering();**

***jButton\_RedigerActionPerformed()***  
Ved klik på knappen jButton\_Rediger, tager den det valgte ordre objekt og kalder OrdreSalg klassens constructor med ordre objektet. Derefter lukker den sig selv og skifter card til ordresalg så kunden kan redigere ordren, hvilket står nærmere beskrevet i beskrivelsen af ordresalg klassen.

***jButton\_VisFakturaActionPerformed***Ved klik på knappen jButton\_VisFaktura åbnes jDialog\_SøgFaktura med fakturaen der er tilknyttet ordren. Hvis der ikke er en faktura tilknyttet ordren er knappen disabled med tooltiptext om ,at der ikke er tilknyttet en faktura til orden.

## Opgrader ordre til faktura og rediger ordre i JDialog\_SøgOrdre

I JDialog\_SøgOrdre har vi et cardlayout hvor vi har flere paneler i som vi bruger til at opgradere en ordre til en faktura og redigere i en eksisterende ordre.

private void opretNormalOrdre()

Metoden bruger vi i opgrader til faktura hvis der ikke er valgt en bedemand. Den tager det validerede data fra inputfelterne i Opgrader card’et og og opretter et faktura objekt med det, hvorefter programmet forsøger at oprette fakturaen i databasen. Den kan også tage et faktureringsadresse objekt og oprette det sammen med hvis dette er valgt.

**private void opretBedemandOrdre()**

Metoden gør det samme som opretNormalOrdre(), men udover de normale faktura data oprettes eller vælges der også en bedemand, der oprettes en provisionsseddel og til sidst oprettes et kontoudtog.

**private void setRedigerInfo()**

Metoden tager det ordre objekt som ligger i JDialogen og udskriver de relevante data fra objektet i GUI’en så man senere kan redigere i det. Den tjekker også hvilke data der er i ordren, f.eks. om der er gravsteninfo i den og viser eller gemmer elementer i GUI’en for at brugeren ikke kan redigere i felter de ikke må.

**private void jButton\_rediger\_acceptActionPerformed(java.awt.event.ActionEvent evt)**

Metoden tjekker først om inputtet I TextFields er validt. Hvis det ikke er udskrives der fejlbeskeder i GUI’en, og metoden slutter. Hvis inputtet er validt åbnes en accept dialog hvor man skal bekræfte at man vil redigere ordren med de data man har skrevet i JDialogen. Når man så trykker accept prøver metoden at redigere ordren i databasen.

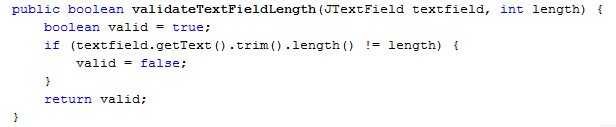
## Fejlhåndtering

Vi har valgt at fokusere meget på exceptionhandling og udskrive beskrivende fejlbeskeder som er overskuelige og står steder i GUI’en der giver mening. Samtidig har vi lagt meget vægt på brugervenlighed, og derfor har vi også lavet validering på alt input der bliver indtastet når vi opretter og redigerer varer, fakturaer og ordre. Disse valideringer fungerer også som exceptionhandling da vi undgår en masse fejl der kunne være opstået hvis brugeren af systemet havde prøvet at oprette og redigere diverse ting med invalid input. Hvis inputtet ikke er validt vil udskriver vi fejlbeskeder ved de enkelte TextFields der har forkert input. De næste eksempler på validering er fra vores OrdreSalg panel, da det er der der er mest validering af input og det er derfor også der det er delt mest op. I nogle af vores andre GUI’er er valideringen ikke så kompleks så der har vi ikke har brug for den samme mængde af metoder til valideringen.

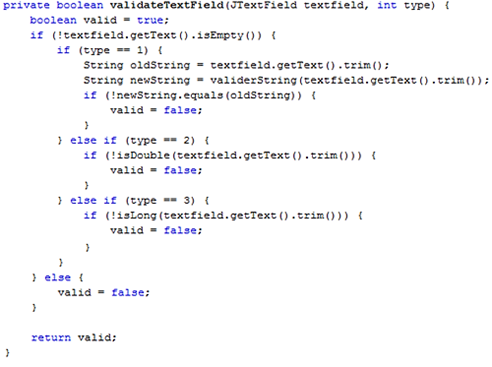
#### isDouble(String input) og isLong(String input)

Metoder isDouble() og isLong() er de meste basiske metoder vi har til validering. De bruges til at tjekke om en String fra f.eks. et TextField er hendholdsvis et heltal i isLong() og et decimaltal i isDouble(). Måden metoderne gør dette på er at vi prøver at parse de strings som metoderne er kaldt med til det input som vi skal bruge, og hvis det kan lade sig gøre returnerer de true og hvis kaster parsen en exception som så bliver fanget og returner false i stedet. Grunden til vi bruger long i stedet for integer til at validere heltal, er at vi har numre som er længere end en integer kan være, f.eks. kontonummer som er på 10 cifre.

#### public boolean validateTextFieldLength(JTextField textfield, int length)

Metoden bliver kaldt med et TextField og en integer som er længden som inputtet skal være. Metoden tager teksten fra TextField’et og bruger trim()metoden som fjerner alle mellemrum der er i starten eller slutningen af den pågældene String. Derefter tjekker den om længden er den samme som den metoden blev kaldt med. Hvis den ikke er returnerer den false og hvis den er returnerer den true. 

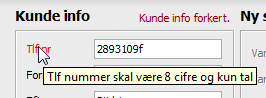
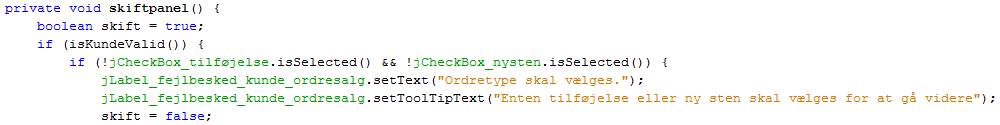
#### private String validerString(String string)

ValiderString metoden tjekker om der er backslash i det input som metoden bliver kaldt med, da et backslash kan ødelægge vores SQL-strenge når vi sætter ind i databasen. Den tager den gamle String og erstatter alle backslashes med ingenting og returnerer den derefter.

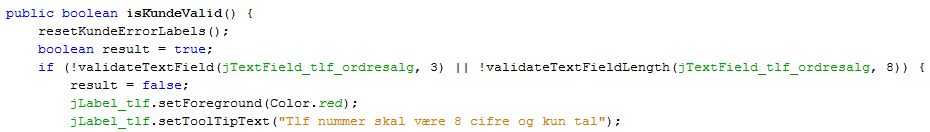
#### public boolean validateTextField(JTextField, int type)

ValidateTextField bruger flere af de ovennævnte metoder til at validere et TextField. Først tjekker den om TextField’et er tomt, som vil sætte TextField’et til invalidt. Hvis der er noget i TextField’et vil koden fortsætte til næste del hvor metoden tjekker hvilken type metoden er kaldt med. Hvis type er 1 tjekker metoden om der er backslashes i String’en, hvis type er 2 tjekker den om String’en er en double ved at bruge isDouble() metoden, og hvis type er 3 bruger den isLong() metoden til at tjekke om det er en long. Hvis valideringen er succesfuld returnerer metoden true og hvis ikke returnerer den false.

#### Eksempel på validering

Et eksempel på den validering der er beskrevet foroven er hvis vi prøver at gå videre til næste del af OrdreSalg ved at trykke på videre knappen i nedre højre hjørne. Hvis f.eks. telefonnummeret ikke er validt vil der Tlf\_nr labelen blive rød og der vil komme et tooltip frem hvis man holder over det hvor der står hvad fejlen er. Derudover kommer der også en general fejlbesked ved kundeinfo panelet så kunden kan lokalisere hvor fejlen er sket henne.

Videre knappen kalder metoden skiftpanel() som starter med at prøve at validere kunden med isKundeValid() metoden. Hvis kunden ikke er valid vil koden ikke gå videre til næste side.

isKundeValid() metoden der kaldes fra skiftpanel() tjekker hvert enkelt TextField i kundeinfo panelet, ved hjælp af validateTextField() og validateTextFieldLength() og hvis de ikke er valide sætter den de fejl i GUI’en som skal vises.

#### Plinger

  
Plinger kan ligesom backslashes ødelægge en SQL-streng da databasen vil opfatte en normal plinger som slutningen af et enkelt input, og derfor laver vi en replace() på alle Strings før de sættes ind i objekterne. Denne replace erstatter en plinger med en plinger med en backslash foran som fungerer som en escape character, så databasen forstår at den plinger der står ikke er en slutning på et input, men derimod et tegn der skal indgå i inputtet.

#### SQLExceptions

Vi har også steder i koden hvor vi fanger SQLExceptions, men da vi allerede har valideret alt data der skal sættes ind i databasen kan fejlene ikke være et problem med vores input. Den eneste fejl der derfor kan være sket når koden fanger SQLExceptions, er en kritisk fejl med databasen, f.eks. hvis der mangler en tabel eller der helt mangler forbindelse til databasen. Derfor har vi valgt at udskrive disse fejl så der står at det er et kritisk problem så brugeren forstår at det er et specielt problem som kræver en løsning eksternt.

#### ControlExceptions

Vi har selv oprettet en klasse der hedder ControlException som bare er en klasse der extender en Exception hvor man selv kan sætte en besked. Vi bruger den i DatabaseObjectHandler klassen til at kaste nye fejl når der er ugyldige data i forbindelse med indsættelse og redigering af databasen.



Her fanger vi exceptionen og skriver at det er ugyldigt objekt data der er problemet og udskriver den detaljerede besked på tooltippet.



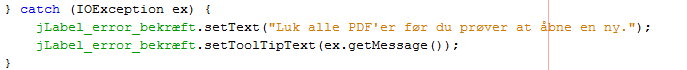
Et eksempel på en ControlException. Her prøver vi at oprette en ordre med et givent ordrenummer, men hvis ordrenummeret allerede findes vil den kaste denne exception.

#### DocumentExceptions

Denne exception kan opstå i sammenhæng med vores oprettelse af PDF’er, men da vi har valideret alt vores data, og vores klasse fungerer fint, er det kun uforventede exceptions der kan opstå og vi har derfor valgt at udskrive fejlen der kan opstå på samme måde SQLExceptions.

#### IOExceptions

Denne exception kan også opstå når vi bruger PDF’er. Den kommer når vi prøver at redigere i en PDF der allerede er åben. Dette burde være den eneste fejl der kan opstå de steder vi bruger den, da vi kun bruger den i forbindelse med oprettelse af PDF’er.



## Test

### Testoversigt

### Generelle forudsætninger:

For at udføre alle tests, er det en forudsætning at:

1. At der er forbindelse til databasen

2. At følgende scripts: 1\_entiteter.sql, 2\_konvertering.sql, 3\_data.sql, er kørt.

3. At computeren har et standard program liggende til åbning af PDF filer.

4. At man er logget ind med korrekte oplysninger: Bruger: test, Password: 123

5. At man starter tests i hovedmenuen.

# JUnit test

Vi har valgt også at oprette junit test i programmet og blackbox teste vores DatabaseObjectHandler klasse. Det ermeget begrænset, hvor mange test vi har, kunne nå rent tidsmæssigt at oprette så det er blevet meget generelle metoder der er blevet testet.

Konklusion på JUnit test.

Selvom vi har relativt få test i forhold til klassens størrelse, opdagede vi en vigtig fejl i rediger kunde. Det er ikke muligt at opdatere telefon nr på en kunde fordi det er den vi identificerer kunden med dvs vi siger ændre tlf nummer på kunde WHERE tlf = detnyetlf. Så vil den aldrig kunne ændre det, da den så vil kigge efter en kunde med det nyenummer i stedet for det gamle.

# 

# It i organisationen

## Bejerholm Stenhuggeri ApS, Organisation og it

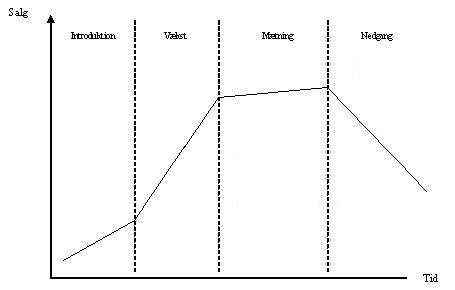
Afsnittet har til formål at give bedre indblik i Bejerholm Stenhuggeris nuværende forretningssituation og arbejdsgang samt at beskrive hvor virksomheden befinder sig i vækstfasen for bedre at kunne forstå hvilke konsekvenser et nyt it system vil kunne have for virksomheden.

### Beskrivelse og arbejdsgang

Bejerholm Stenhuggeri er en lille virksomhed beliggende i Korsør. Virksomheden blev grundlagt i 2010 og valgte efter en god opstart, at udvide i 2011 med en detailforretning i Kalundborg. Bejerholm Stenhuggeri håndterer salg af et sortiment af stenprodukter, som hovedsageligt er bestående af gravsten.

Virksomheden køber stenplader fra forskellige leverandører, hvoraf Bejerholm selv producerer gravsten, fuglebade og efter bestilling, andre stenprodukter. I sjældne tilfælde købes der færdigt producerede gravsten til videresalg. Virksomheden yder også service i form af rensning og oppudsning af sten.

Det unge stenhuggeri, har altså både produktions-, service- og til dels handels aktiviteter. Bejerholm Stenhuggeri kan derfor betegnes som værende en kombination af en produktionsvirksomhed, en servicevirksomhed og til dels en handelsvirksomhed.



*PLC Kurve/Livscyklus*

Ovenstående PLC kurve illustrerer en virksomheds livscyklus. I den første fase, fødselsfasen, som Bejerholm Stenhuggeri befinder sig i, er virksomheden domineret af ejeren. Det er Rene Bejerholm der tager alle beslutninger vedrørende virksomheden. Det er ligeledes ham der producerer varerne samt står for indkøb og salg. Han etablerer ligeledes forbindelsen til samarbejdspartnere og leverandører.

I fase 2, vækst-fasen, vil virksomheden blive delt op i mindre afdelinger. Vækster Bejerholm Stenhuggeri yderligere, vil Bejerholm ikke kunne deltage i alle virksomhedens aktiviteter eller tage stilling til alle problemer som måtte opstå undervejs.

i modenhedsfasen vil organisationen være nået et punkt, hvor den fungerer stabilt. Udviklingen af nye produkter kan aftage, og i nedgangsfasen kan markedsandele begynde at falde og virksomheden så småt begynde at lukke med mindre der sker eller er sket, en fornyelse af virksomhed.

Rene Bejerholm, indehaver, har siden virksomhedens opstart, ikke haft et IT system til registrering af salg eller håndtering af varelager. Når en kunde bestiller en vare, skrives ordren ned i hånden. Hver morgen bruger Bejerholm derefter en til to timer, på at skrive ordre samt fakturaer ind i et dokument, hvorefter han printer dem ud og sender dem per post til kunderne. Derudover har optælling af varelager til regnskabet ligeledes været en tidskrævende arbejdsopgave, idet alle gamle fakturaer skal læses igennem og skrives ned. Rene står derudover for alle virksomhedens arbejdsopgaver, som består af: salg og indkøb, fakturering, ledelse, administration, produktion samt udvikling af forretningsprocesser. Dog med hjælp fra hans søn Mathias, og en ansat til at tage imod kundeordre i butikken i Kalundborg.

#### Forretningskoncept og vision

Rene Bejerholm ønsker, som de fleste små virksomheder, at udvide sin virksomhed på sigt. Som tidligere nævnt, har han da også allerede udvidet virkomheden med detailforretning i Kalundborg.

" *Jeg har været mange år i branchen som ansat. Målet med vores egen virksomhed, Bejerholms Stenhuggeri, har altid været et højt serviceniveau, med kvalitet i top, og priser i bund* ". Tirsdag den 13. september 2011, 09:19, Ugeavisen for Kalundborg Kommune & omegn.

Som den lille lokale virksomhed Bejerholm Stenhuggeri er, lægges der altså vægt på værdier som service og priser. Måden hvorpå Bejerholm skaber værdi for hans kunder er altså gennem gode oplevelser, service samt en konkurrencedygtig pris. Bejerholm er bevidst omkring vigtigheden af støtte og opbakning af det lokale samfund. Han vil hellere "*sende en kunde hjem, som er i for stor sorg, til at tage en beslutning om, hvilken gravsten der skal vælges, end at sælge den for derefter at få en kunde som er ked af det produkt personen har fået*". Rene Bejerholm forstår altså forholdet til nærmiljøet, i dette tilfælde, kunder og konkurrenter er essentielt for en virksomheds eksistensgrundlag.

#### Leawitts diamant

For at kigge nærmere på virksomheden og hvordan delene i virksomheden pårvirkes af hinanden, er Leawitts diamant inddraget. Modellen består af fire grundvariable; struktur, mennesker, opgaver og teknologi. Sker der ændringer i en variabel, påvirker det de andre og omvendt. Der er tilføjet variablene nærmiljø samt fjernmiljø idet udefra kommende faktorer også spiller ind i en virksomheds virkelighed.

**Nærmiljø**

**Fjernmiljø**

#### Mennesker:

Eftersom Rene Bejerholm tager sig af alle forretningopgaver, og kun har en ansat i Kalundborg er der to aktører i virksomheden. Bejerholm Stenhuggeri er dermed meget skrøbelig i sin uværende form, idet Rene stort set er den eneste som håndterer virksomhedens arbejdsopgaver. Dette medfører dog også at virksomheden er meget fleksibel og hurtig vil kunne omstille sig.

Struktur:

Arbejdsdelingen er meget simpel, idet kun Rene Bejerholm og en anden er ansat i virksomheden. Fordelingen af ansvar og myndighed er begrænset og ændringer i opgaver, struktur samt opgaver vil derfor påvirke Rene i højeste grad.

#### Opgaver:

som tidligere nævnt består opgaverne af produktion af produkter, varer og tjeneste ydelser, indkøb, produktion, fakturering og salg.

#### Teknologi:

Bejerholm har de fornødne maskiner til rådighed, som kræves for at producere hans sten.

Dog er IT en del som Bejerholm har manglet. Som nævnt tidligere, har arbejdsgangen været præget af dobbeltarbejde, blyant og viskelæder i form af ordre- samt fakturabehandling. Idet der intet system har været til at håndtere varelageret, skal alle vare tælles op og alle gamle ordre og fakturaer læses igennem. Bejerholm har altså haft begrænsede tekniske hjælpemidler, i form af et program, som kan hjælpe ham med at effektivisere hans arbejdsgang. Manglen på dette har altså påvirket både virksomhedens struktur, opgaver samt mennesker.

#### IT systemet og konsekvenser

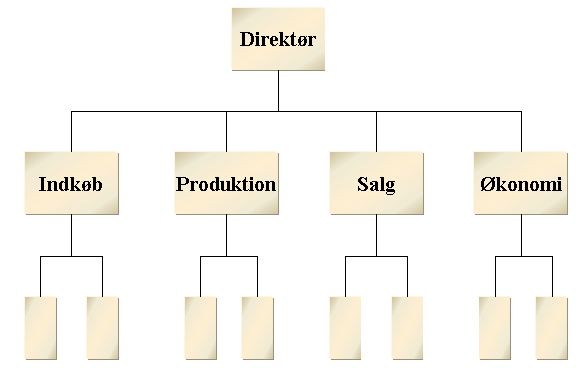
Befandt Bejerholm Stenhuggeri sig i en sen vækstfase eller modenhedsfasen i forhold til PLC kurven, ville forandringen i teknologi variablen kunne have større effekt på de øvrige variable; struktur, opgaver samt mennesker/aktører. Man skulle da være opmærksom på, at der kan være større konsekvenser ved implementering af et nyt IT system. Uddannes personalet fx ikke gennemtænkt i brug af systemet, vil dette kunne have negative konsekvenser i forhold til de ansattes udførsel af deres arbejdsopgaver.

Omvendt ville en succesrig implementering af systemet have stor værdi for virksomheden. Da det udviklede system håndterer ordrer, fakturaer samt status på varelageret vil de arbejdsopgaver som før var en tidskrævende proces, spare Bejerholm for ca. 60 timers arbejde pr måned. Dermed vil Bejerholm kunne skabe mere værdi for hans timer. De timer som ellers ville blive brugt på dobbelt indskrivning og status, vil nu kunne bruges på at producere og sælge mere. Det vil dermed forøge Bejerholms muligheder for at vækste.

Da virksomhedens form er så enkel og størrelsen lille, vil en implementering af et system være mindre krævende og indebære mindre risiko for negative konsekvenser. Da Bejerholm Stenhuggeri er derfor mindre skrøbelig i forhold til at være omstillings dygtig er det vil derfor med fordel kunne implementeres nu.

### IT og vækst

Vækster Bejerholm Stenhuggeri i fremtiden, vil der kunne opstå behov som det udviklede system, ikke vil dække. I takt med at virksomheden bliver større, vil antallet af ansatte og arbejdsopgaver vokse, og organisationsstrukturen vil blive mere kompleks.

I de fleste virksomheder som Bejerholm Stenhuggeri, er funktionsprincippet det mest anvendte i forhold til hvilket arbejdsdelingsprincip, der er gældende. Dette vil betyde at ansatte har ansvar og myndighed alt efter hvilken funktion i virksomheden deres arbejdsopgaver afdækker.

For at disse kan snakke sammen, vil der for eksempel opstå et behov for en udvidelse af systemet således at det håndterer produktionen bedre, ligesom Bejerholm selv har nævnt at han i fremtiden også ønsker at systemet skulle kunne indeholde funktioner til bedre at kunne lave regnskaber. Systemet er nu opbygget således, at det ikke håndterer hvor langt en vare er fra at være færdigproduceret. Bejerholm kan lave udtræk af varelagerets værdi, men økonomidelen er meget begrænset. Som beskrevet tidligere i Systemudvikling, blev det da også overvejet om programmet skulle kunne håndtere dette, men det blev vurderet ikke at være nødvendigt idet Bejerholm ønsker at slette en stenplade på lageret og tilføje den som ny vare, når denne er færdigproduceret. Idet det kun er Bejerholm der håndterer processen fra salg, til produktion og levering, vil det derfor ikke være en funktion som ville blive brugt. I fremtiden ville systemet derfor kunne udvides, således at man kunne holde styr på hvor langt en vare var i processen fra at være klar til levering og udvide funktioner til håndtering af virksomhedens økonomistyring. Der vil i nedenstående afsnit redegøres for hvad et ERP system er.

## ERP systemet

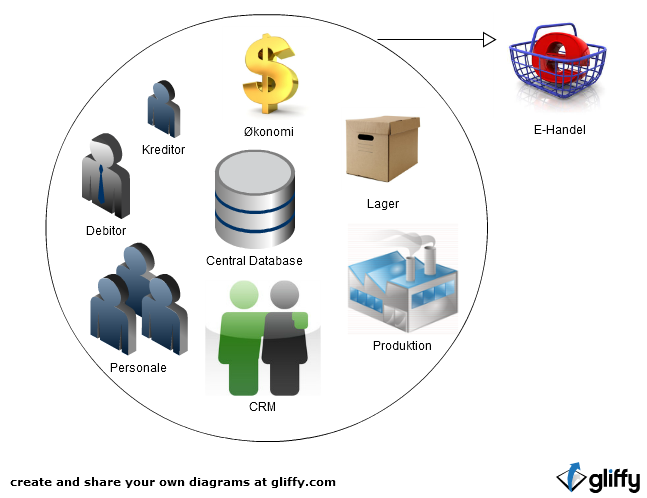
Dette system er en udvidelse af virksomheden og skal betragtes som en løsning for virksomheds arbejdsbelastning. ERP systemet er til formål ved udvidelse af Bejerhoms Stenhuggier at gøre medarbejdere mere produktive i takt med virksomhedens vækst. Denne løsning er til for at komme videre fra grundlæggende regnskabsføring med forbedret regnskabsfunktionalitet. Det vil stryke væksten i virksomheden med mindre påvirkning af lønningslisten og it-budgetter. Derfor vil dette være en optimal løsning, under væksten for virksomheden.

ERP - Enterprise Ressource Planning. Hvad kan man bruge systemet til i virksomheden, og hvilke fordele/ulemper er der ved implementeringen af sådan et system? Da det er et mere komplekst spørgsmål kræver det eftertanke om hvad det skal betyde for virksomheden, og systemets karakteristika og funktioner.

ERP defineres ud fra to synsvinkler:

Systemets tekniske karakteristika og funktioner.

Systemets forretningsmæssige betydning for virksomheden.

”ERP – Systemer er informationssystemer, der understøtter online, integreret, realtime transaktionensregistrering i forbindelse med afviklingen af virksomhedens forretningsprocesser ved hjælp af en central database, ofte i en client/server IT-arkitektur.”

ERP består af følgende som vist på billedet.

Når man implementer et system, skal det være en investering der kan ses med markante forbedringer på.

En rigtig ERP-løsning for din virksomhed skaber målbar merværdi i alle områder fra implementering til drift. Arbejdet med at finde den rigtige it-løsning kræver ikke stor videnskab, men kræver dog grundighed, indsigt og visioner.

Før man tager en beslutning, bør man som minimum have klarhed omkring:

Forretningen – hvordan forretning drives i dag og hvad vil man ændre.

Målsætningen – hvad ønsker man at opnå, og hvilke behov skal opfyldes for at det kan se.

Kompetencer – Har man brug for ny viden, og hvordan sikre du man sig den.

Hvor mange ressourcer du skal bruge på det indledende analysearbejde afhænger af virksomhedens størrelse og kompleksitet.

Overvejelser, når målet er defineret og visionen klar.

Når man har det klare øjebliksbillede, dukker der muligvis nogle spørgsmål op, som man kan have brug for hjælp til at besvare:  
- Skal der købes nyt eller optimere eksisterende løsning?  
- Hvilken løsning opfylder vores behov bedst?  
- Hvem kan hjælpe os sikkert i mål?

Når man har dannet sig en forståelse for kravene kan man først der handle med idéerne.

I vores situation drejer det sig om en mindre virksomhed, Bejerholms stenhuggeri ApS. Virksomheden er styret af en mand, og hvis han integrer sin virksomhed til en ERP løsning, vil alt hans papir arbejde, lige fra faktura til revisor seddel stort set blive overtaget af ERP virksomheden, så han kun skal koncentrerer sig om at hugge løs og ellers tage i mod kunder så de kan se på den sten de nu vil have. På den måde kan det være en af mange situationer, hvor det kan løse problemet hvis de på forhånd har behandlet ordren over internettet. Men hovedsagligt tager ERP sig for alle former af ordrebehandling, så han ikke behøver at stå og tænkte på det, når han i stedet kan fokuser på kunden.

Fordele:  
- En mere effektiv arbejdsgang  
- Styr på arkiveringen  
- Undgå papirarbejde  
- Flere sten pr dag  
- Bedre service

Ulemper:  
- Mindre virksomhed  
- Omkostninger kontra omsætning

Det er spiller ind, om det kun er bogholderi der inddrages eller flere løsninger så som transport. Der er ERP-systemer der tilbyder transport hjælp, og det kunne være en hovedløsning til virksomheden efter som det er store sten der er tale om, som kommer til at skulle transporteres frem og tilbage, og muligvis en ekspedering af transport området, så når der bliver evt. solgt dekoration til haven at der en leveringsmulighed, VISMA er en af mange der tilbyder dette.   
  
Der er flere aspekter, når man taler om en from for udbyggele eller ekspedering, om det er Globalisering, eller en lokal renovering. Men man skal også passe på det ikke pludseligt går forstærk så virksomheden ikke kan følge med, da det kan være en virksomheds under gang. Derfor kan det være en god ide med ekstern hjælp, hvis man vil gå så langt så er et ERP system en god ide at gøre brug af.

## Organisation og It-sikkerhed

### Hvorfor har vi valgt:

Vi har valgt at inddrage organisation og it-sikkerhed som et af hovedemnerne i IT i Organisationen. Beslutninger kommer på baggrund af at systemet er et meget data baseret system, herunder er der mange faktorer der gør sig gældende i og omkring den sikkerhedsmæssige beskyttelse af disse data. Samtidig er det også et emne hvortil der er rig mulighed for fremtidige opdateringer til det nuværende system med henblik på at forbedrede sikkerheden endnu mere.

### IT-Politik.

Det nyudviklede system til Bejerholm Stenhuggeri, skal udskifte den tunge arbejdsgang med papirarbejde og arkivering af store mængder data. Det er derfor vigtigt, at der er tiltro til programmet fra brugerens side. Herunder, at alle de data der trækkes ud fra databasen til systemet, om en given faktura/ordre/kunde på et givent tidspunkt er de korrekte data. Hele virksomheden afhænger af at systemet udregner de rigtige priser, udskriver fakturaen til den rigtige kunde, og holder styr på igangværende fakturaers betalingsstatus mm. Ydermere er det helt essentielt at systemet er stabilt, især på baggrund af den tilstand de kunder, der betjenes under oprettelsen af orden er i. Der er ikke plads til at systemet går ned når der trykkes på godkendt ordre, hvorefter kunden så skal igennem hele proceduren igen.

### Logisk Sikkerhed

Da systemet varetager håndteringen af alt data, er det vigtigt, at det ikke er muligt for uautoriseret brugere at tilgå systemet. Der er derfor opsat et loginsystem, som kommer frem ved program start, og beder om brugernavn og kode. Først efter korrekt indtastning kommer man videre til den del af systemet, hvor man kan se/ændre/slette data.   
Det er muligt for brugeren ”Bejerholm” at oprette, redigere og slette andre brugere, men alle andre brugere har kun mulighed for at ændre deres egen kode. På den måde undgår man at Bejerholm selv bliver låst ud af systemet af en fremtidig ansat.

Det er ikke muligt i systemet, at se en historik over handlinger en bruger har foretaget sig, men en fremtidig implementering kunne indeholde sådanne mulighed. En implementering til at begrænse brugerroller ville også finjustere datasikkerheden således at bestemte brugerroller kun kan oprette - og redigere ordre, men ikke slette dem eller ændre betalingsstatus. Ydermere kunne en handlingshistorik være relevant hvis system får yderligere brugere så man kunne se en administrativ liste over de handlinger der er foretaget i system og af hvilken bruger.

## Fysisk sikkerhed

Computeren står placeret i et ikke aflåst kontor ved siden af hovedindgangen til butikken. Det er her ordre bliver oprettet sammen med kunden. Som en sikkerhedsforanstaltning skal der minimum være en kode til computeren, således at der ikke er adgang til selve computeren. Samtidig ville login efter at aktivitet på computeren har været inaktiv i en periode være at foretrække. Man logger nemlig kun ind 1 gang når systemet startes, herefter bliver man ikke spurgt igen. Så som en ekstra sikkerhed kunne systemet påkræve en kode efter 10 minutters inaktivitet, så man undgår uautoriseret brug af systemet.  
Som et led i at beskytte brugerdata ved tyveri, kunne det også være en fordel hvis de gemte pdf filer som bliver oprettet i systemet ikke bliver gemt lokalt på computeren, men i stedet ha en netværksharddisk eller server som de blev gemt direkte på. Denne enhed skulle så placeres et sikkert sted i bygningen således at filerne er utilgængelige for fremmede såsom indbrudstyve. Samtidig kunne en wire fastsat til computer og borde eller lign. Hjælpe med at besværliggøre eller helt stoppe indbrudstyven i at stjæle computeren. Til sidst skal alt it-inventar sikres med beskyttelsesmærkning således afsætningsværdien af det stjålne falder drastisk og gør det nemmere at identificere ved en eventuel efterforskning eller tilbageleverings handling.

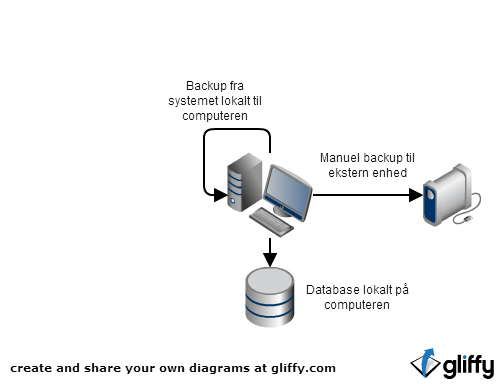
I tilfælde af strømsvigt, computer defekt eller andre lignende hændelser. Skal ordreseddel og fakturaseddel være tilgængelig i papirform. På den måde lider virksomheden ikke noget økonomisk tab ved sådanne et tilfælde. Efter genopretning tastes de udfyldte skemaer ind i system og er derved opdateret igen.

### Datasikkerhed

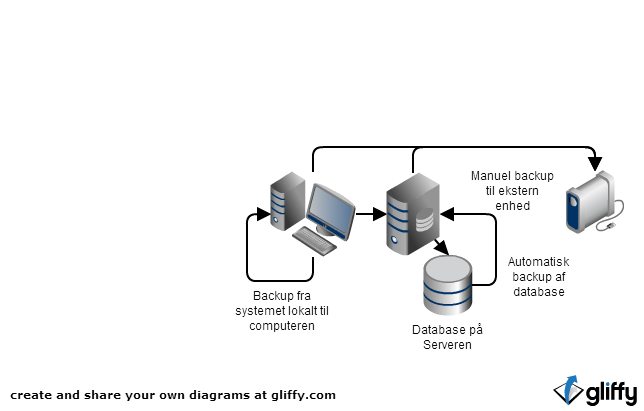
Butikkens hovedbutik er i Korsør, og det er også her computeren med systemet er placeret.   
Selvom systemet er lavet til at være et enkelt brugersystem, er det stadigvæk muligt at kunne have fjernadgang til selve databasen. Derfor ændrer datasikkerheden sig alt efter hvilken database løsning der bliver implementeret.   
  
Man kan opstille 3 konkrete scenarier:

**Databasen installeres lokalt på computeren**:

Denne løsning er ikke anbefalelsesværdig i og med alt for mange faktorer spiller ind med hensyn til at sikre tilgængeligheden at dataene. Det vil skabe unødvendigt arbejde for virksomheden, da de vil være tvunget til at skulle foretage backup langt oftere for at sikre sig, at data ikke går tabt. Alligevel vil en harddisk fejl, anden hardwarefejl eller tyveri være fatalt, såfremt man ikke har fået lavet en ekstern backup og derfor ikke har mulighed for at genskabe databasen. Selv ved ekstern backup kan en ordre være gået tabt, som derefter skal genskabes igen i systemet manuelt. Det vil kort sagt kræve for mange arbejdstimer og ricisien er for stor, skulle man i en længere periode ikke have fået taget backup.

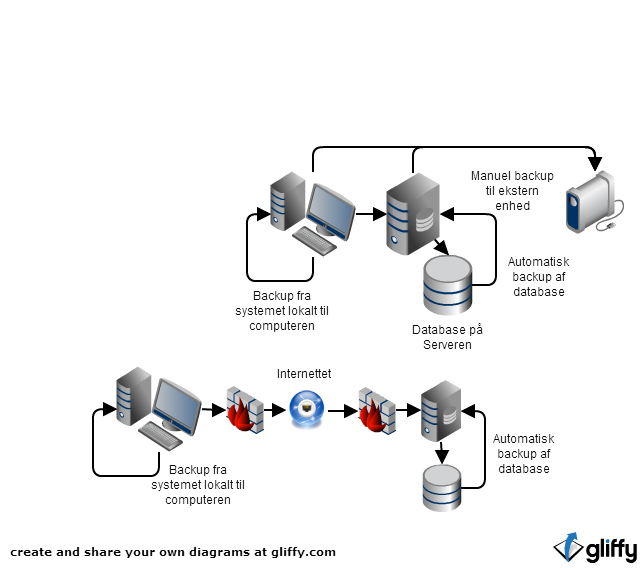


**Databasen installeres på en server og tilgås via lokalnetværket.**En server sættes op et sikkert og aflåst sted i butikken, hvor der dagligt automatisk laves backup af databasen. Efter endt uge kunne der tages en separat backup fra systemet i tilfælde af at serveren skulle stå af. Ved denne løsning slipper firmaet for selv at skulle foretage den daglige backup. Ved computer nedbrud vil det være nemt og hurtigt at installere programmet igennem også forbinde det til databasen som ligger på serveren med alt data intakt. Som en ekstra sikkerhed kunne man lave et brandsikret rum/område til serveren således tilgængeligheden af virksomhedens data er sikret ved sådan en hændelse. Ved denne løsning er det vigtigt også at sætte fokus på kommunikationssikkerheden således at uvedkommende ikke får adgang til firmaets data udefra. Hvis computeren er tilsluttet internettet er derfor vigtigt at computeren er opdateret med de seneste opdateringer samt at der er installeret antivirus program. Ydermere skal routeren der forbinder computer og internet været sikret med firewall. En lokal firewall på computeren ville også være at foretrække som en ekstra sikkerhed.



**Databasen installeres på en online server og tilgås via internettet:**

Denne løsning fokuserer meget på en eventuel opgradering af systemet samt database, så de ville kunne håndtere flere brugere af gangen. På den måde ville man kunne have en computer i hver butik, som ville køre programmet og hente data fra samme database. Løsningen kræver også at man er ekstra opmærksom på at alle sikkerhedskrav er opfyldt. Herunder kryptering af data samt alle computere er opdateret, med tilhørende sikkerhedsværtøjer som nævnt ved det andet scenario.



### Persondataloven

Systemet indeholder bevidst ikke Cpr-nummer da det ikke er relevant for Bejerholm at vide ved køb af varer i butik eller ved bestillingsarbejde. Til gengæld bliver der lagret andre informationer om kunden såsom navn, adresse, by og telefon nr. Samt bank oplysninger for bedemændene, som der er samarbejde med. Det forventes derfor at disse informationer er fortrolige mellem firma og kunde, og derfor er det vigtigt at disse data er beskyttet og kun tilgås af autoriseret personer. Ved deling af data mellem computerne i butikken er det vigtigt det sker forsvarlig. Hvis det er en USB drevet lageringsenhed skal den beskyttes med kode og/eller dataene krypteres såfremt den skulle falde i de forkerte hænder. Det samme gælder ved online kommunikation alt data skal krypteres så det ikke opfanges af en 3 part.

# Slutevaluering

Tests

Ordre redigering

f

Rykker - kan ikke sendes.

Kunde data kan ændres og dermed ændres gamle ordre og faktura også