Afsluttende eksamensopgave

Titel: EARSS Data Management Program

IMM – B. Eng – 2007 - 9

Afleveringsfrist: Fredag d.12. januar 2007

s032244, Olsen, Stefan Schytte _____

Denne opgave indeholder __ sider inkl. denne side, samt 1 CD

Forord

Denne rapport er skrevet for at dokumentere mit arbejde med mit eksamensprojekt "EARSS Data Management Program".
Projektet har varet 10 uger, og er udført i perioden 30. oktober 2006 til 12. januar 2007. Det er lavet på - og for Statens Serum Institut (SSI) på Amager, i Afdelingen for Antibiotikaresistens og Sygehushygiejne (AAS).
I fire måneder op til projektets start har jeg været i praktik i samme afdeling, hvor jeg har haft mulighed for at sætte mig ind i opgaven og indsamle det nødvendige data til selve eksamensprojektet. Intet af det arbejde der indgår i eksamensprojektet er lavet i praktikperioden, men er alt sammen lavet i de 10 uger der er afsat til eksamensprojektet.
Rapporten er skrevet i et sprog der henvender sig til studerende på samme niveau som jeg selv, men uden kendskab til den aktuelle opgave.
Dvs. at jeg går ud fra at læseren er bekendt med de fra undervisningen kendte begreber, og jeg vil derfor ikke forklare hvad de er, men i stedet koncentrere mig om hvordan jeg har brugt dem.

Projektbeskrivelse

European Antimicrobial Resistence Surveillance System (EARSS) er et europæisk projekt under EU kommissionen, der overvåger antibiotikaresistens, hos bakterier, i Europa.
De første antibiotika blev udbredt omkring 2. Verdenskrig, som behandling mod bakterieinfektioner. Allerede få år efter så man de første eksempler på bakterier der var blevet resistente, altså modstandsdygtige, overfor visse typer antibiotika. Det betød at man ikke længere kunne behandle disse infektioner med de normale antibiotika.
Siden er resistensen steget støt og roligt, og i dag er flere antibiotika stort set uden virkning. Heldigvis fremstilles der nu mange forskellige typer antibiotika der virker forskelligt, så man har stadig et godt forsvar overfor bakterier. Men da man ikke kan være sikker på at man i fremtiden hele tiden kan finde nye typer antibiotika, er det nødvendigt at prøve at begrænse resistensen.
Skrækscenariet er at man til sidst ikke har nogle virksomme antibiotika til at behandle visse infektioner. Dette kan i værste fald medføre at man kan dø af hvad vi i dag anser som simple infektioner.
EARSS overvåger som sagt antibiotikaresistens i Europa. Dette gøres ved at man indsamler resistensdata fra de ca. 30 deltagende lande. I hvert af disse lande indsamles resistensdata fra landenes laboratorier centralt og indsendes til EARSS' hovedkvarter, der ligger hos det hollandske institut for folkesundhed og miljø (RIVM), i Bilthoven i Holland.
I Danmark foretages denne indsamling af data af "mit" afsnit, på SSI. I hvert amt er der en Klinisk Mikrobiologisk Afdeling (KMA), og to mere i København. KMA'erne indeholder laboratorier der ligger på nogle af landets største sygehuse. Når læger i praksis og på hospitaler tager patientprøver af f.eks. blod eller urin, for at afgøre om patienterne har en infektion, bliver prøverne indsendt til en KMA. KMA'en analyserer så prøven og laver bl.a. en

resistensbestemmelse, der viser om bakterien er modstandsdygtig overfor visse antibiotika.

1474	2005	25914	B	240259	FRMESM5	20050302	20050303	20050307	10000	10000	K	23000	36400	S
1475	2005	25914	B	240259	FRMESM5	20050302	20050303	20050307	10000	10000	K	23000	36460	S
1476	2005	26210	B	140263	ASMMMM1	20050303	20050304	20050308	10000	10000	M	23000	12600	S
1477	2005	26210	B	140263	ASMMMM1	20050303	20050304	20050308	10000	10000	M	23000	13000	R

Eksempel på data indsendt fra en Klinisk Mikrobiologisk Afdeling i Excel-format.

EARSS indsamler resistensdata for syv forskellige bakterier. De deltagende lande indsender data for så mange de har mulighed for, og i Danmark er der hidtil blevet indsendt resistensdata for to typer bakterier, nemlig stafylokokker og streptokokker. Disse data kommer fra to af SSI's egne laboratorier, der arbejder med disse to bakterier, og er ikke indsamlet elektronisk, fra de enkelte laboratorier rundt om i landet.
I år 2006 blev der så også indsendt elektronisk indsamlede data for E. coli bakterier, men kun fra 5 KMA'er. Planen er fremover at udvide indsamlingen til alle 7 bakterier, men da KMA'erne indsender data frivilligt må vi tage det lidt ad gangen, for ikke at overbelaste dem.
Data for E. coli er blevet indsamlet i elektronisk form, og det er meningen at al indrapportering skal foregå sådan fremover.
Det er meningen at KMA'erne skal indsende de samlede resistensdata til SSI i elektronisk format, foreløbig én gang om året, men da denne måde at samle data ind på først er begyndt i 2006, er dataindsendelsen stadig lidt tilfældig.
KMA'erne bruger ikke alle de samme laboratoriesystemer til at behandle deres data. De bruger i øjeblikket 4 forskellige systemer, men tallet kan ændres, da et par KMA'er overvejer at skifte system indenfor en overskuelig fremtid.
Disse systemer eksporterer hverken resistentdata i samme format, eller fil-format. Der eksporteres både i xml-filer, Excel-filer og tekst-filer. De forskellige værdier hedder heller ikke det samme, eksempelvis hedder en *E.coli* bakterie "eco" i ét system, og "23000" i et andet. Der er heller ingen garanti for at to KMA'er der bruger samme system sender ens filer, da de

hver især har mulighed for at ændre værdier i systemet, give dem nye navne osv.
Da EARSS skal have data i et bestemt format og værdierne i bestemte koder, er der derfor behov for at oversætte KMA-data, og udskrive dem i det rette format i en tekstfil, der kan sendes til EARSS.

```
20060613 DK016 25914 bl 20050302 240259 2 2 1959 016G 1 med eco AMC S
20060613 DK016 26210 bl 20050303 140263 1 2 1963 016A 1 med eco AMC I
```

Samme data som i figuren ovenfor, men oversat til EARSS-format.

Min opgave har derfor været at lave et system der kan læse disse forskellige fil-formater, oversætte dem til EARSS-formatet og udskrive igen, i EARSS-format.
Idéen om at indsamle data elektronisk på denne måde blev startet for flere år siden og en tidligere medarbejder udviklede en Visual Basic 6 applikation der kunne læse XML-filer fra Næstved KMA.
Tiden er dog løbet fra denne applikation af flere årsager. Dels fordi den byggede på en hjemmelavet XML-parser, der ikke kører specielt hurtigt, og dels fordi der er blevet tilføjet en del til XML-filerne fra Næstved siden det blev lavet. Desuden kunne applikationen ikke læse andre filer end dem fra Næstved, hvilket var med til at gøre den ubrugelig.
En løsning kunne have været at opdatere denne applikation. Men da der er sket meget siden VB6, og der skulle laves en stor del om, samt tilføjes endnu mere, har det været nemmere at starte fra bunden med et helt nyt program, som jeg har valgt at lave i C# og .NET 2.0.

Valg af værktøjer

Til løsning af opgaven har jeg haft frie hænder hvad angår programmeringssprog og værktøjer.
Jeg har dog valgt imellem Open Source programmer, for ikke at bruge afdelingens penge på ting der fås stort set lige så gode gratis.

Af programmeringssprog har jeg valgt at bruge C# på .NET 2.0 platformen, da jeg synes det er et godt sprog at arbejde med, og jeg har brugt det i tidligere kurser. Der er mange gode funktioner i og desuden arbejder en af de andre på kontoret også med det.
Som IDE har jeg valgt at bruge SharpDevelop Version 2.0.0. Hovedsageligt fordi den er gratis men også fordi den er overskuelig og forholdsvis nem at lave brugergrænseflader i.

Det data der er blevet konverteret på den "gamle" måde, er blevet opbevaret i Excel-filer og tekst-filer. Indtil nu er der kommet én fil om året fra ganske få KMA'er. Målet er at modtage en fil fra hver KMA pr kvartal, hvilket hurtigt vil gøre det uoverskueligt, hvis man opbevarer det i enkelte filer.
Derfor har jeg besluttet at basere programmet på en relationel database, der vil samle al data ét sted, og gøre det væsentligt mere overskueligt og nemmere at lave de samlede udtræk til EARSS.
Jeg har også selv kunnet vælge hvilken database jeg vil bruge.
Jeg ville umiddelbart have valgt at bruge MySQL databasen, da jeg har brugt den ved flere lejligheder i diverse projekter.
En af fordelene ved MySQL databasen er, udover at den er gratis, at der var lavet en udmærket front-end til den, der gjorde det nemt at oprette og administrere tabeller, forespørgsler osv. Desværre har der været nogle uoverensstemmelser mellem folkene bag MySQL og dem der lavede front-end'en, der har gjort at front-end'en er blevet trukket tilbage og ikke længere er til at få fat i.

Jeg har derfor valgt at bruge PostgreSQL databasen, der minder meget om MySQL og har en indbygget front-end, med de samme funktioner som dem jeg brugte i den anden. Der har derfor ikke været nogen mærkbar forskel ved at skifte til PostgreSQL databasen.

Analyse

Krav til programmet

Set fra brugerens synsvinkel kan kravene til programmet beskrives simpelt:

1. Man skal kunne indlæse resistensdata fra en fil.
2. Man skal kunne udtrække data igen i EARSS-format

Der er dog lidt mere i det end som så.
Data skal kunne indlæses fra forskellige filformater og laboratoriernes egne koder skal oversættes til EARSS' koder. De KMA'er der eksporterer til XML filer bruger alle det samme program, nemlig et program der hedder MADS. De eksporterer alle i det samme XML format, men bruger ikke nødvendigvis de samme koder.
Helt så heldigt står det ikke til med de resterende KMA'er. De eksporterer alle i enten Excel eller almindelige tekst filer. Tekstfilerne er enten komma- eller Tab-separede og kan derfor med fordel importeres i Excel, da det gør læsningen af filerne væsentlig lettere. Men her stopper ensartetheden også. Der er nemlig ingen fælles standard for rækkefølgen, eller antallet af, værdierne. Da denne måde at indsamle data på er så ny, og da der, fordi KMA'erne deltager frivilligt, er grænser for hvor meget arbejde vi kan pålægge dem, kan det heller ikke garanteres at to filer fra samme KMA vil have ens format. Det skulle dog være sikkert at de koder hver enkelt KMA bruger er de samme hvilket betyder at det kun er antallet og rækkefølgen af værdierne der er et problem.
Derfor er det et vigtigt krav til programmet at man kan indlæse filerne selvom rækkefølgen af værdierne er ukendt.
Når det kommer til udtrækket af data til EARSS er der også visse krav.

Da det ikke er hele databasen der skal udtrækkes og sendes hver gang skal man kunne vælge hvilken periode man vil have data fra. Eksempelvis at man vil have alle resultater fra år 2005. Nogle KMA'er indrapporterer også resistensdata for andre bakterier end dem der skal sendes til EARSS. Dette kan være nyttigt i forbindelse med antibiotikaresistensovervågningen her hjemme, så selv om EARSS ikke vil have disse data er der en god grund til at gemme dem i databasen alligevel. Det er derfor et krav at man kan vælge hvilke bakterier man vil udtrække data for, således at EARSS kun får hvad de beder om.
Desuden er der et "lidt løst" krav til hastigheden hvormed datafilerne læses, da det jo var en af årsagerne til at det tidligere program ikke længere fungerer så godt. Grunden til at det er "lidt løst" er at der ikke er noget specifikt mål for hvor hurtigt det skal gå, men det er jo altid en fordel hvis man ikke skal vente længere end højst nødvendigt på at få indlæst sine data.

EARSS fil-formatet

Når EARSS modtager en datafil fra et land skal data først kontrolleres og derefter lægges i den centrale database. Som nævnt skal data leveres i et bestemt format, for at EARSS kan læse det. I alt kan der indrapporteres 29 forskellige felter om hver resistensbestemmelse til EARSS, men kun 14 er obligatoriske. I Danmark beder vi i øjeblikket kun KMA'erne om at indsende de obligatoriske, for at gøre det så nemt for dem som muligt. Hvis de vil indsende flere er de selvfølgelig velkomne, men indtil videre holder de sig til de obligatoriske.
Data skal indsendes i en almindelig tekstfil. Værdierne skal listes i en bestemt rækkefølge med én resistensbestemmelse i hver linje, dvs. op til 29 værdier i hver linje. Landende kan selv vælge mellem to måder at skille værdierne ad på.
Den simpleste måde er at lave en TAB efter hver værdi. Den anden måde er at hver værdi har en bestemt længde. Her listes værdierne lige efter hinanden, og hvis en værdi ikke har den rette længde må man bruge blank-tegn til at fylde ud. Det er ikke muligt have værdier der er længere end det specificerede. EARSS ved så f.eks. at prøve-id'et starter ved karakter 18 og slutter ved karakter 30, og kan på denne måde skelne alle karaktererne der ellers står som en lang tekststreng ud i et.
Jeg har lagt en liste over værdierne, taget fra EARSS manualen 2005, i bilaget "**Data Exchange Format**".

Usecases

I dette afsnit har jeg vha. usecases beskrevet funktionerne i programmet. Brugeren af systemet vil hovedsageligt være den nationale EARSS data manager, men kan i princippet være hvem som helst. Jeg vil derfor benævne vedkommende "brugeren".

Usecase nr. 1

Navn: Indlæs fil fra KMA

Formål: At få lagt data fra den pågældende KMA i databasen.

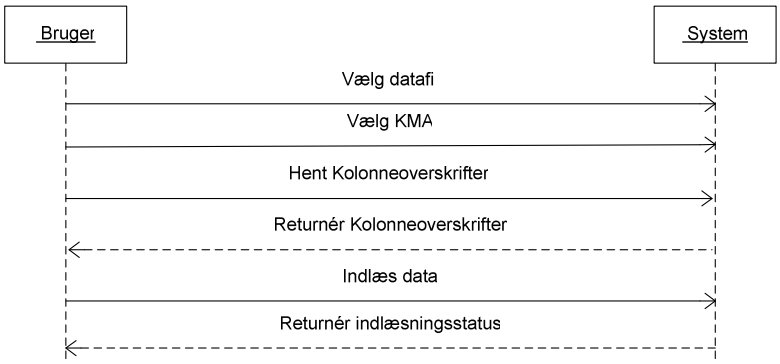
Prækondition: En KMA har indsendt en datafil, der skal indlæses og lægges i databasen.

Succes scenarium: Data bliver lagt i databasen.

Hovedforløb:
1. Brugeren vælger fanebladet "Indlæs Data"
2. Brugeren browser frem til den pågældende fil
3. Brugeren vælger hvilken KMA filen kommer fra
4. Brugeren trykker på knappen "Hent overskrifter"
5. Brugeren vælger hvilke kolonner de forskellige værdier skal hentes fra.
6. Brugeren trykker på knappen "Indlæs data"
7. Dataindlæses i databasen.

Alternativt forløb:
4a. Filen er en XML-fil fra et MADS-system og der skal derfor ikke vælges kolonner. Gå direkte til punkt 6.
7a. Der opstår en fejl under indlæsningen, og brugeren får en fejlmeddelelse herom. Data bliver ikke lagt i databasen.

Postkondition: Data er lagt i databasen.



Usecase nr. 2

Navn: Udtræk data til EARSS

Formål: At udskrive en tekstfil indeholdende data til EARSS

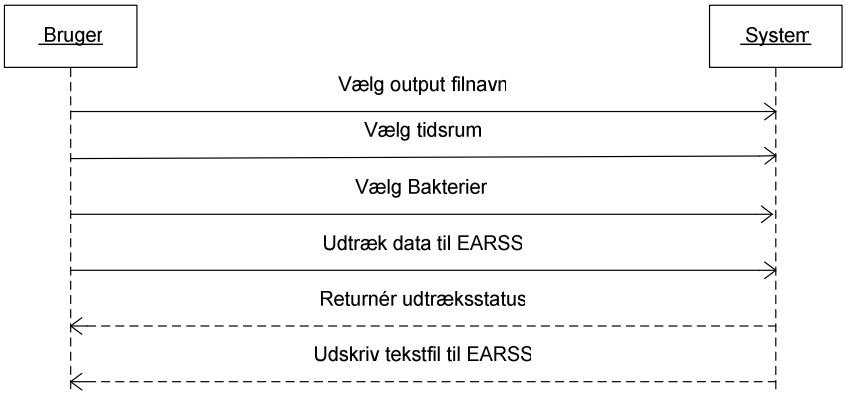
Prækondition: Der skal indsendes data til EARSS, og disse skal udtrækkes fra databasen.

Succes scenarium: De ønskede data bliver udskrevet i en tekstfil.

Hovedforløb:
1. Brugeren vælger fanebladet "Udtræk Data til EARSS"
2. Brugeren vælger hvilken periode der skal udskrives data for.
3. Brugeren vælger hvilke bakterier der skal med i udtrækket.
4. Brugeren skriver det ønskede navn på output filen.
5. Brugeren trykker på knappen "Udtræk data"
6. Data udskrives i en tekstfil

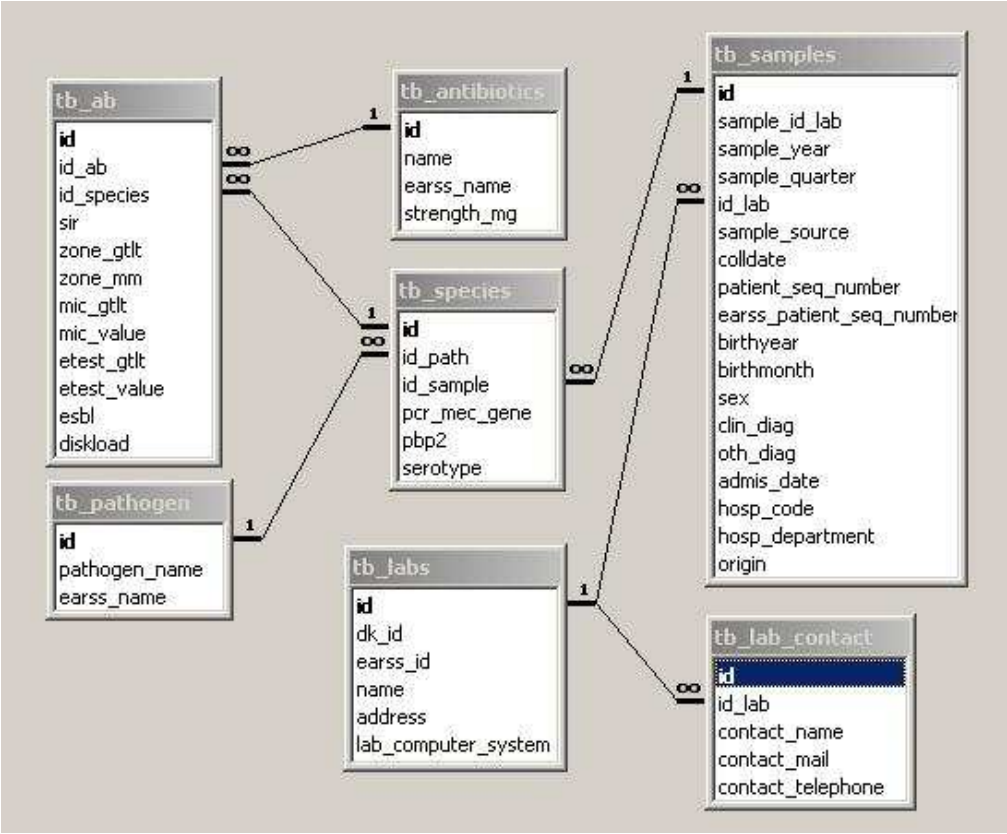
Alternativt forløb: 6a. Der opstår en fejl under udskrivningen, og brugeren får en fejlmeddelelse herom. Data bliver ikke udskrevet

Postkondition: En tekstfil med data til EARSS er udskrevet.



Databasen

Til implementering af databasen har jeg som sagt valgt at bruge PostgreSQL database systemet.
Databasen er først og fremmest designet til at indeholde de nødvendige data til EARSS. Der udover indeholder den også lidt ekstra informationer, f.eks. navn og adresse på KMA'erne, der ellers bare ville være repræsenteret i databasen ved en EARSS kode. Ved fremtidige udvidelser vil det være nyttigt at have disse oplysninger i databasen.

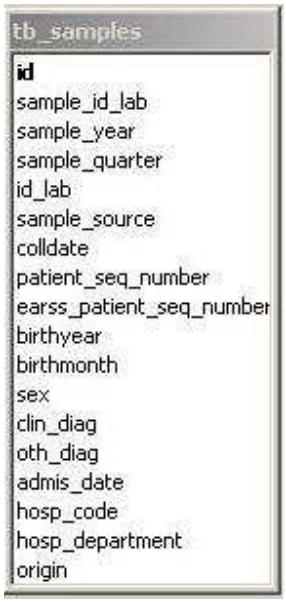


ER-diagram over databasen.

Databasen består af 7 tabeller der tilsammen afspejler de prøver der rapporteres til EARSS. Det data der udtrækkes til EARSS er placeret i tabellerne **tb_samples**, **tb_species** og **tb_ab**, der kan ses på ER-diagrammet

ovenfor. De fire andre tabeller **tb_antibiotics**, **tb_labs**, **tb_lab_contact** og **tb_pathogen** indeholder supplerende oplysninger om de forskellige antibiotika, bakterier, KMA'er og KMA'ernes kontaktpersoner.
Jeg har valgt at give hver tabel i databasen en automatisk optalt integer som primærnøgle. Jeg kunne, som en del af databaselitteraturen gør, have sammensat primærnøgler af flere felter i hver tabel, men synes ikke at det tjente noget formål, men tværtimod komplicerede det unødigt. Eksempelvis ville en unik nøgle i tb_samples have krævet KMA-id'et (id_lab), KMA'ens eget sample id (sample_id_lab), året (sample_year) og muligvis kvartalet (sample_quarter), afhængigt af hvor ofte KMA'erne nulstiller deres fortløbende sekvensnumre. Det mener jeg ville skade mere end det gavner.

Prøvedata er struktureret i tre niveauer. Prøveniveau, bakterieniveau og antibiotikaniveau. På øverste niveau er data knyttet til selve prøven, f.eks. en blodprøve der er taget for at undersøge om patienten har bakterier i blodet, der ikke skal være der. Disse data har jeg lagt i tabellen **tb_samples**. Det er oplysninger om hvor og hvornår prøven er taget,
hvad det er for en type prøve (blod, urin, spinalvæske eller andet).
Tabellen indeholder også data på patienten, som alder, køn, indlæggelsesdato og eventuelle diagnoser. Disse patientdata ville jeg normalt have lagt ud i en tabel for sig, men har i dette tilfælde ikke gjort det. Det skyldes at jeg ikke kan være sikker på om den samme patient optræder flere gange under forskellige patient sekvensnumre. Normalt ville man jo bruge en form for "kunde ID", til at identificere de enkelte patienter med, som f.eks. CPR-nummeret. De fleste KMA'er bruger også CPR-nummeret som primærnøgle på deres patienter, men da det er person-følsomme oplysninger der er tale om, er det ikke alle KMA'er der vil indsende hele CPR-nummeret, på trods af at SSI har tilladelse fra



datatilsynet til at behandle disse oplysninger. Det medfører at patienten med sekvensnummeret "1234" med stor sandsynlighed ikke er den samme som har nummer "1234" i den næste datafil fra den KMA.
Den anden vej rundt kan man eller ikke udelukke at den samme patient optræder flere gange i samme datafil, men med forskelligt sekvensnummer. Jeg vil derfor ikke få nogen fordel ud af at trække patienten ud i en tabel for sig, da jeg vil skulle oprette en ny for hver prøve alligevel, og derfor har jeg beholdt patientdata i tb_samples.
Da nogle KMA'er sender det fulde CPR-nummer som patient sekvens nummer, og dette ikke må videregives til EARSS har jeg også oprettet et earss_patient_seq_number, således at begge numre kan gemmes i databasen. Det er dog ikke sikkert at CPR-nummeret vil bliver gemt i databasen, pga. regler vedr. opbevaring af CPR-numre.
Feltet 'id_lab' i tb_samples, er en fremmednøgle, fra tabellen **tb_labs**. tb_labs indeholder oplysninger om KMA'erne.

Hver KMA har et dansk id, et id hos EARSS, et navn som f.eks. Aalborg KMA, en adresse og et computer system.
Normalt når det drejer sig om adresser ville jeg lægge postnummer og by ud i en tabel for sig selv jf. 3. Normalform. I dette tilfælde vil det dog ikke give nogen mening at oprette en postnummertabel. Det skyldes at der kun er 15 KMA'er, og de befinder sig i hver deres amt, og dermed også hver deres postnummer. KMA'erne er de eneste der opbevares adresser på, og der er derfor ikke andre tabeller der vil kunne nyde godt af en separat postnummertabel. Og som fremtiden ser ud, men de nye regioner, vil der ikke komme flere KMA'er, snarere tværtimod.
Hver KMA har mindst en kontaktperson, men da nogle har flere end en, har jeg lagt kontaktpersonerne ud i en tabel for sig selv, tabellen **tb_lab_contact**. Oplysningerne der gemmes på kontaktpersonerne er navn, e-mail adresse og

telefonnummer.
En prøve kan indeholde fra nul til mange forskellige bakterier. Men KMA'erne indsender kun data på de prøver der indeholder mindst én bakterie. En post i tb_species kan kun tilhøre én post i tb_samples, og derfor er forholdet mellem tb_samples og tb_species én til mange.

Tabellen **tb_species** indeholder data på bakterieniveau. Hver prøve indeholder som oftest kun én bakterie, men kan som sagt indeholde flere bakterier og data på hver type bakterier opbevares i denne tabel.

Tabellen indeholder to fremmednøgler, id'et på den prøve den tilhører i tb_samples (id_sample), og id'et på den type bakterie den indeholder, som kan findes i tabellen tb_pathogen. Felterne pcr_mec_gene, pbp2 og serotype er alle tre oplysninger om bakterien, der ikke er interessante i databasesammenhæng.
tb_pathogen tabellen indeholder bakteriens navn, samt den kode EARSS bruger for bakterien.

Som nævnt bliver det data der udtrækkes til EARSS opbevaret i tabellerne tb_samples, tb_species og tb_ab. De resterende fire tabeller tb_pathogen, tb_labs, tb_lab_contact og tb_antibiotic indeholder supplerende data til brug i fremtidige udvidelser af programmet. Det er hovedsageligt det rigtige navn på bakterier, antibiotika osv., der giver mere mening end EARSS-koderne hvis data skal præsenteres og læses af mennesker.

Tabellen **tb_ab** indeholder det tredje og laveste niveau, der er resistensbestemmelserne for hvert enkelt antibiotikum, på hver enkelt bakterie.

Hver bakterie kan man teste for dens resistens mod flere forskellige antibiotika, og hver resistens bestemmelse kan kun tilhøre én post i tb_species, hvilket er årsagen til at forholdet mellem tb_species og tb_ab også er én til mange.
I én sætning kan man sige at en prøve (tb_samples) kan indeholde én eller flere bakterier (tb_species) og hver bakterie kan være testet mod én eller flere antibiotika.

tb_ab indeholder feltet id_ab, der er fremmednøgle fra tabellen tb_antibiotics. tb_antibiotics indeholder oplysninger om de forskellige antibiotica, der ligeledes vil kunne bruges i fremtidige udvidelser til programmet.

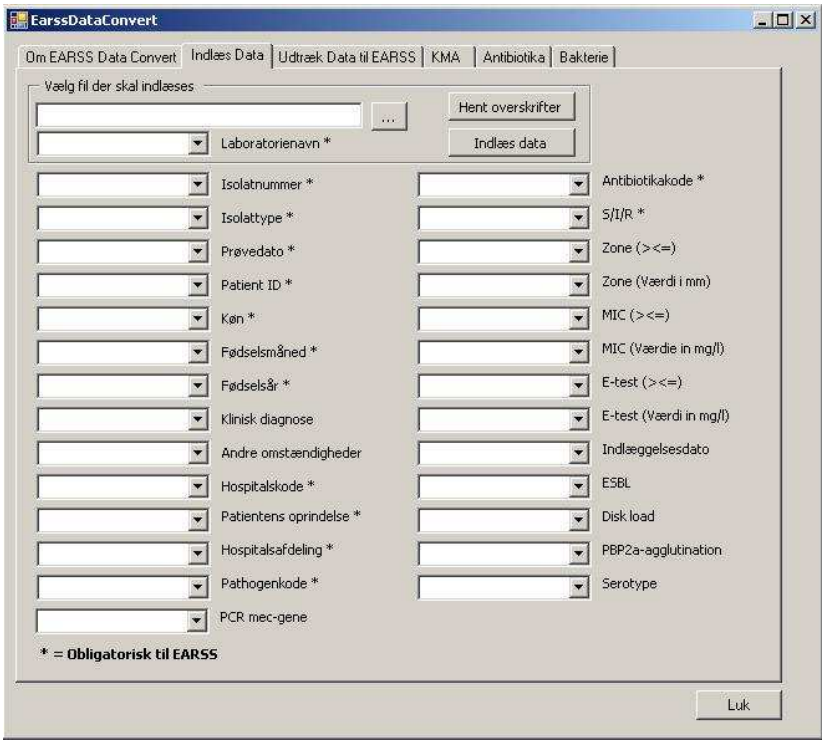
Design

Brugergrænsefladen

Jeg har valgt at opbygge brugergrænsefladen vha. faneblade, da jeg synes det er en god og enkel måde at få god plads på sin brugergrænseflade på. Når programmet starter vises det første faneblad "Om EARSS Data Convert", der kan ses på forsiden af rapporten. Her vises en kort beskrivelse af programmet, og hvad det kan.

Andet faneblad, "Indlæs Data", bruges når en datafil skal indlæses.
Som nævnt i opgavebeskrivelsen bruger KMA'erne fire forskellige laboratoriesystemer. Hvidovre, Herlev, Slagelse og Aalborg bruger ADBakt systemet der er lavet af svenske Autonik.
Skejby, Rigshospitalet, Næstved, Esbjerg, Vejle, Herning, Viborg og Odense bruger MADS-systemet, der er udviklet på Skejby Sygehus.
Roskilde Sygehus får analyseret sine prøver Statens Serum Institut, der bruger et system der hedder LIMS.
Endelig bruger Hillerød Sygehus SafirLIS systemet.
Bortset fra MADS-systemet, der eksporterer XML filer, eksporterer alle de andre laboratoriesystemer i enten Excel- eller tekstdokumenter. I tekstdokumenterne er værdierne enten kommaseparerede eller tabseparerede. Det gør det nemt at importere dem i Excel, der er væsentlig nemmere at læse fra, da man der har fordelen af at kunne indeksere efter række- og kolonnenummer.

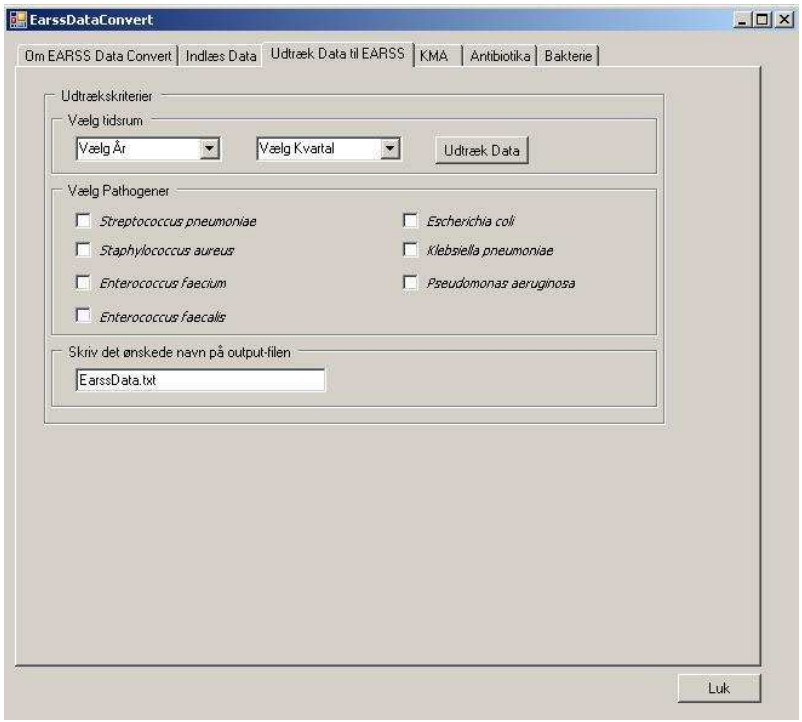
De fleste KMA'er sætter en overskrift på kolonnerne i deres datafiler. På de filer der ikke måtte have overskrifter er det nemt at tilføje en ekstra række i toppen af Exceldokumentet og give kolonnerne en overskrift. Disse overskrifter bruger jeg til at fortælle programmet hvilke værdier der står i hvilke kolonner. Når brugeren har valgt den fil der skal læses, vælges filens afsender fra combo-boxen "Laboratorienavn". Hvis der vælges et laboratorium der bruger MADS-systemet bliver alle de resterende combo-boxe samt knappen "Hent overskrifter" blive deaktiverede. Hvis der vælges et andet laboratorium er det et Exceldokument der skal læses og combo-boxene og knappen "Hent overskrifter" forbliver aktive. Ved at trykke på "Hent overskrifter" hentes alle overskrifterne fra Exceldokumentet ind, og indlæses i alle combo-boxene. Hver combo-box svarer til en EARSS-værdi, og man kan på denne måde vælge hvilke kolonner de forskellige værdier skal hentes fra. På denne måde løses problemet med at finde ud af rækkefølgen på værdierne i Excel- og tekstdokumenterne. Når kolonnerne er valgt trykkes på "Indlæs data" for at starte indlæsningen.



Som sagt eksporterer MADS-systemet data i XML-filer. MADS-systemet er for nylig blevet opdateret så det nu kan eksportere de nødvendige data til EARSS i et fast format der blev fastlagt i efteråret 2005. Den nye opdatering

er i øjeblikket ved at blive udbredt til alle KMA'er der bruger MADS-systemet, hvilket medfører at alle kommer til at indsende data i det samme format. Jeg kan derfor være sikker på at alle XML-tags har de samme navne lige meget hvor XML-filen kommer fra. Det betyder at det ikke er nødvendigt at fortælle programmet hvor det skal finde værdierne hver gang, men at det kan gøres en gang for alle i selve programmet.

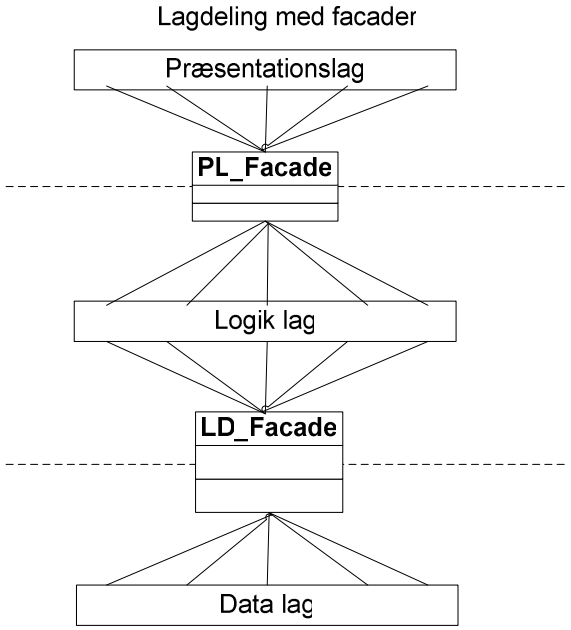
Tredje faneblad på formen bruges til at udtrække data til EARSS i en tekstfil. Når der skal udtrækkes data skal brugeren vælge hvilken periode der skal udtrækkes data fra, f.eks. 1. kvartal 2005.



Dette gøres ved at vælge året og kvartalet i de to combo-bokse "Vælg År" og "Vælg Kvartal". Derpå vælges hvilke bakterier der skal udtrækkes data for. Dette gøres ved at krydse af i check-boksene ud for bakteriernes navne.
Til sidst skrives det ønskede navn på output-filen og der klikkes på "Udtræk Data".
De tre sidste faneblade indeholder i øjeblikket ikke noget. De er sat på da jeg har tænkt mig med tiden at udvide programmet til også at kunne opdatere databasen vedrørende KMA'er, antibiotika og bakterier. Dette er i øjeblikket ikke så aktuelt da der ikke umiddelbart er nogle ændringer på vej indenfor nogle af disse områder.

Opbygningen af programmet

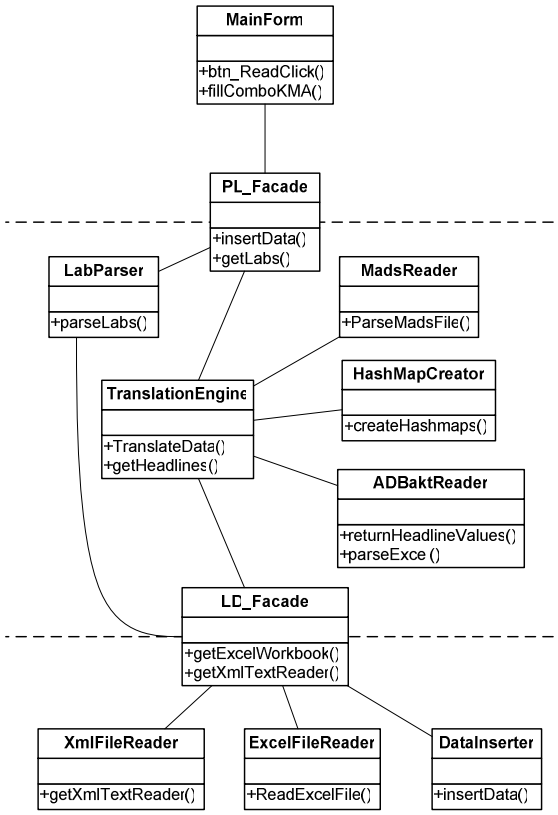
Jeg har valgt at bygge programmet op i tre lag. Et lag til præsentation, altså brugergrænsefladen, et lag til "motoren" eller logikken, og et lag der indeholder alle I/O forbindelser, som f.eks. forbindelsen til resistensdata-filerne, oversættelses-filerne og databasen.
Præsentationslaget indeholder udelukkende brugergrænsefladen og er det mindste lag.
Logiklaget er det lag hvor selve databehandlingen foregår. Her læses og oversættes data, og det er også her det bliver gjort klar til at blive lagt i databasen. Når data skal udtrækkes til EARSS er det også her det bliver gjort klar inden det sendes til dataforbindelseslaget, der skriver filen til harddisken. Alt andet end at tage imod input fra brugeren, og styre skrivning til - og læsning fra harddisken gøres i dette lag.
Dataforbindelseslaget tager sig af alle forbindelser til "omverdenen", bortset fra forbindelse til brugeren, der jo håndteres af præsentationslaget.
Hver gang der skal skrives til eller læses fra harddisken, hvad enten det er i form tekstfiler, Excelfiler, XMLfiler eller databasen, går det gennem dette lag.



Klasserne i dataforbindelseslaget forbinder udelukkende logiklaget med harddisken. Det data der skal læses fra disken bliver læst ind et passende objekt, der så sendes direkte til logiklaget. Hvad der så skal ske med det der er logiklagets "problem". På samme måde er al data der skal skrives til harddisken gjort klar af logiklaget, og datalaget skal bare skrive det til disken.
Jeg har valgt at bruge et facade-pattern til at adskille de tre lag.
Det har jeg valgt fordi jeg synes det er en god måde at adskille lagene på, så programmet er mere overskueligt.
Præsentationslaget og logiklaget bliver adskilt af en facadeklasse der hedder *PL_Facade*. Al kommunikation mellem præsentationslaget og logiklaget går gennem denne klasse, hvilket naturligvis giver lidt mere kode. Dette bliver dog mere end opvejet af den bedre overskuelighed, da alt der bliver sendt mellem de to lag er repræsenteret af en metode i PL_*Facade*.
På samme måde er logiklaget og dataforbindelseslaget adskilt af en facadeklasse der hedder *LD_Facade*. Hver gang et objekt i logiklaget har brug for at hente fra eller skrive til en fil eller databasen kontaktes denne klasse. På denne måde behøver logiklaget ikke kende til klasserne i dataforbindelseslaget, men kan bare sende sine forespørgsler til LD_*Facade* der så fordeler dem videre nedefter.

Dataindsættelse

Når brugeren skal indlæse data i databasen skal han vælge hvilken KMA filen er kommet fra. Jeg har lavet en XML-fil med alle KMA'ernes navne og EARSS-id, der hedder Laboratories.xml. Filen hentes af klassen LabParser, der kan ses i klassediagramudsnittet til højre. Af pladshensyn har jeg delt klassediagrammet op i to, og kun taget de vigtigste klasser med. Et samlet og komplet klassediagram kan ses i bilaget "Klassediagram".



Som nævnt kan man ikke regne med rækkefølgen af værdierne i de filer der ikke kommer fra MADS-systemet. Derfor har jeg lavet det sådan at brugeren selv kan fortælle programmet hvilke værdier der står i hvilke kolonner i Exceldokumentet.
Når brugeren har valgt kolonnerne i combo-boksene og trykket på "Indlæs data" oprettes en hashtabel med navnet på værdierne som nøgle og kolonne indekset som værdi. Når der skal læses fra Excel-arket kan programmet så slå op i hashtabellen eksempelvis med nøglen "pathogen" og få kolonnenummeret på den kolonne der indeholder bakteriekoden tilbage.
Da ikke alle værdier er obligatoriske til EARSS, behøver man ikke at vælge kolonner til alle værdier.
Selve oversættelsen af data styres af klassen *TranslationEngine.* I *TranslationEngine* afgøres det om filen er en XML-fil eller en Excel-fil.

Herefter bliver datafilen hentet ind fra enten klassen *XmlFileReader* eller *ExcelFileReader*, og så skal data oversættes.
Til at oversætte fra KMA'ernes koder til EARSS-koder har jeg valgt at lave en oversættelsesfil i XML-format til hver KMA. Et eksempel på en sådan fil kan ses i bilaget " **Forkortet XML-fil til oversættelse**". Filerne indeholder oversættelser af de værdier der har brug for oversættelser, f.eks. prøvetyper, hospitalskoder og afdelingsspecialer. Filerne hedder det samme som KMA'en + ".xml" som f.eks. "Aalborg.xml", og indeholder desuden KMA'ens EARSS-id og det datoformat KMA'en bruger. Et eksempel fra Næstved.xml filen kan ses i figuren herunder.

```
<lab_id>DK007</lab_id>
<date_format>YYYYMMDD</date_format>
```

KMA-id og datoformat angivet i oversættelsesfilen for Næstved KMA

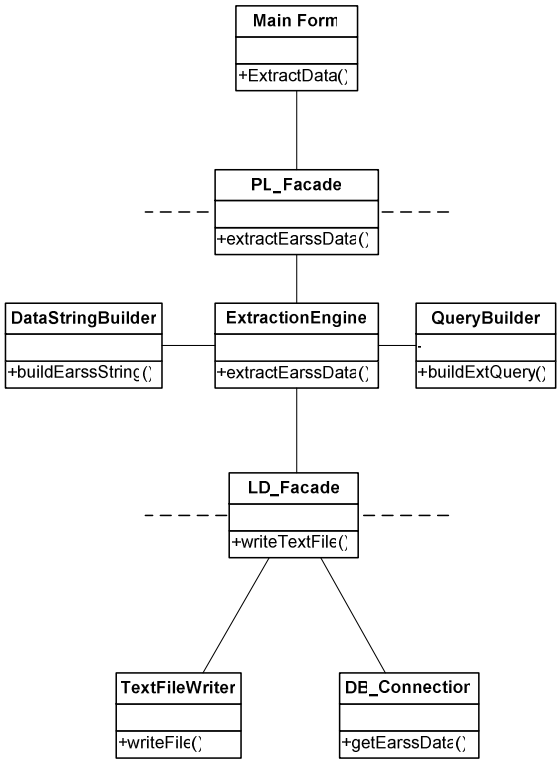
Ikke alle KMA'er bruger datoformatet "yyyymmdd" som EARSS bruger. Derfor er det nødvendigt at fortælle programmet hvilket format der bruges, så det selv kan bytte rundt på år, måned og dag, og skrive datoen i det rigtige format. Som det ses er det dog ikke nødvendigt i eksemplet ovenfor.

Jeg kunne have valgt at lægge disse oplysninger i databasen, i stedet for at oprette selvstændige filer til de enkelte KMA'er. Jeg har dog valgt XML-filerne da oversættelses-oplysninger bruges jævnligt i andre sammenhænge også uden at programmet og databasen kører. I disse tilfælde er det nemt lige at slå op i XML-filerne og finde de oplysninger man skal bruge. Desuden synes jeg at XML-formatet er godt og nemt at arbejde med.
Klassen *HashMapCreator* laver en række hashtabeller, ud fra XML-oversættelsesfilen, som enten sendes til klassen *MadsReader* eller klassen *ADBaktReader*, afhængig af om det er en XML- eller Excel-fil. Hvis det er en Excel-fil sendes også kolonne-hashtabellen til *ADBaktReader*. Selvom

klassen hedder ADBaktReader læser den også Excel-filerne fra SafirLIS, og LIS systemerne. *MadsReader* og *ADBaktReader* sender det oversatte data tilbage til *TranslationEngine*, der sender det til klassen *DataInserter*. Fra DataInserter bliver data lagt i databasen

Dataudtrækning

Udtrækket af data til EARSS bruger, ud over *MainForm* og facade-klasserne, fem klasser. Disse klasser, og de funktioner der bruges undervejs, kan ses på klassediagramsudsnittet til højre. Når brugeren har valgt sine søgekriterier og trykket på knappen sendes søgekriterierne via facaden til *ExtractionEngine*, der styrer selve dataudtrækket. Når data skal udtrækkes skal der laves en forespørgsel til databasen. Denne forespørgsel skal indeholde søgekriterierne, og den bliver lavet af klassen *QueryBuilder*, der returnerer den færdige forespørgsel.



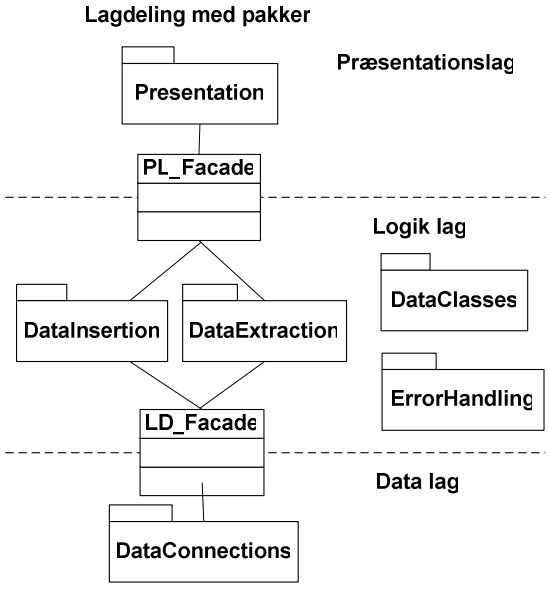
Klassediagram for udtræksdelen.

ExtractionEngine sender derpå forespørgslen via facaden til klassen DB_*Connection*, der forbinder og sender forespørgslen til databasen. Når DB_*Connection* får data retur fra databasen sendes dette tilbage til *ExtractionEngine*.

Data skal nu formateres til EARSS-formatet, og *ExtractionEngine* sender derfor dataet til klassen *DataStringBuilder*, der formaterer data så det stemmer overens med EARSS-formatet, og sender det retur. Data er nu klar til at blive udskrevet, og sendes af *ExtractionEngine*, via facaden, til klassen *TextFileWriter*, der udskriver data i en tekstfil.

Implementering

For nemmere at kunne overskue alle klassefilerne i programmet har jeg delt dem op i pakker, efter deres funktioner. Pakker kaldes i C# namespaces, og svarer til hvad der i Java kaldes packages. I et eller andet omfang hænger alle klasserne jo sammen, men nogle hænger mere sammen end andre, f.eks. dem der har med indlæsning eller dataudtræk at gøre.
I alt er det blevet til syv pakker, **DataClasses**, **DataConnections**, **DataExtraction**, **DataInsertion**, **ErrorHandling**, **Facades** og **Presentation**.



På figuren til højre ses pakkerne, og i hvilket lag de hører til. De to facader er dog vist som klasser, for at det skal give mening på figuren. DataClasses, der udelukkende indeholder klasser til "transport" af data rundt i programmet og ErrorHandling der logger eventuelle fejl i systemet, bruges i alle lagene, og hører derfor ikke specielt til i Logiklaget.
Indholdet af de resterende pakker forklarer nogenlunde sig selv.
Jeg har valgt at lade dataindsættelsen og – udtrækningen blive styret af henholdsvis en TranslationEngine og en ExtractionEngine. Disse to klasser styrer indsættelsen og udtrækket ved at sende data rundt til de klasser der oversætter det, sende det til databasen osv., og modtage det igen når det er blevet behandlet af en klasse. Jeg synes dette er en pænere og mere overskuelig løsning end alternativet, der ville have været at sende det gennem en længere række klasser der kunne manipulere data, eller endnu værre at lade al manipulationen foregå i én klasse.

Dataindsættelse

Når brugeren trykker på "Indlæs Data"-knappen, der i virkeligheden hedder btn_Read, oprettes kolonne-hashtabellen *ht*. Kun hvis filen er en Excel-fil fyldes noget i *ht*. Når programmet startes i *MainClass* opretter denne formen, der hedder *MainForm*, med en instans af *PL_Facade*. Via denne instans sendes *ht*, filnavnet og KMA-id'et til en instans af klassen *TranslationEngine*, der som tidligere nævnt styrer indlæsning og oversættelse af data. *TranslationEngine*, bruger i en switch KMA-id'et til at afgøre hvilken type til der skal indlæses. Når det er afgjort startes oversættelsen i en ny switch, der styres af filtypen.

```
switch (fileType){
    case "xml": // Håndtér XML fil

    case "xls": // Håndtér Excel fil
}
```

Forsimplet udgave af måden hvorpå der skelnes mellem filtyperne.

Hvis filen er en XML-fil, oprettes en instans af MadsReader-klassen og datafilen hentes ind, via *XmlFileReader*-klassen og *LD_Facade*. Herefter kaldes funktionen *createHashmaps* i en instans af klassen *HashmapCreator*. *createHashmaps* indlæser og parser KMA-oversættelsesfilen, og opretter en række hashtabeller med de forskellige typer værdier i, som f.eks. prøvetyper, antibiotika osv. Dette gøres i fire "nestede" switch-sætninger. Hashtabellerne lægges i en instans af klassen *TranslationHashTables*, der udelukkende bruges til at sende disse hashtabeller rundt i programmet i. Til sidst returneres instansen af *TranslationHashTables* til *TranslationEngine*.
Nu er alt klar til at oversættelsen kan begynde, hvilket gøres i funktionen *ParseMadsFile* i *MadsReader*-klassen. Her parses datafilen og de værdier der hører til på Sample-niveauet lægges i et Sample-objekt, de værdier der hører til på Species-niveauet lægges i et Species-objekt og de værdier der hører til på Ab-niveauet lægges i et Antibiotic-objekt. Antibiotika-objekterne lægges i

en ArrayList hos det Species-objekt de hører til, og Species lægges i en ArrayList i det Sample-objekt de hører til. Endelig lægges alle Sample-objekterne i en ArrayList, der hedder *translatedData*. Inden værdierne lægges i disse objekter bliver de der har behov for det oversat ved hjælp af opslag i hashtabellerne fra *TranslationHashTables*-objektet.
Nogle værdier skal også ændres som f.eks. datoer der ikke er i det rette format eller kvartalet der skal aflæses ud af måneden prøven er taget i. Dette gøres i en instans af klassen *ReadDatafileToolbox*, der indeholder en række funktioner til dette formål.
Når al data er læst ind, oversat og lagt i ArrayLister, returneres *translatedData* til *TranslationEngine*.
TranslationEngine sender nu det oversatte data via *LD_Facade* til en instans af klassen *DataInserter*. *DataInserter* er undtagelsen der bekræfter reglen, hvad angår min lagdeling, og at data kun bliver behandlet i Logik-laget. Når data skal indlæses skal der sendes mange SQL-forespørgsler af sted til databasen, og der kommer også en del svar tilbage. Derfor synes jeg at det er bedst at have afsenderen af alle disse forespørgsler så "tæt" på databasen som muligt, uden at skulle sende forespørgslerne ned gennem *TranslationEngine* og *LD_Facade* hver gang.
InsertData-metoden i *DataInserter* benytter sig af såkaldte Prepared Statements til at indsætte data i databasen. Da der skal indsættes data i tre tabeller, tb_samples, tb_species og tb_ab, har jeg lavet Insert-forespørgsler, der bliver oprettet i hver sin NpgsqlCommand, nemlig insertSample, insertSpecies og insertAb.

```
insertSpecies = new NpgsqlCommand("INSERT INTO tb_species" +
                                  "(id_pathogen, id_sample) " +
                                  "VALUES( :pathogen,:sampleid)",conn);
```

Prepared Statement til indsættelse af Species

Hvor der normalt er Values i SQL sætningen er der her variable, der kan ses med et kolon foran. Inden selve indsættelsen går i gang gøres de tre Prepared

Statements klar ved at tilføje parameternavne og typer, og lige inden "det går løs" kaldes *prepare()*-metoden på dem, så de nu er klar til at blive eksekveret. Eksekveringen foregår i tre foreach-løkker, en for hvert af de tre niveauer i resistensdataet.
Den første løber alle samples i ArrayListen translatedData igennem, sætter parametrene og eksekverer insertSample. Herefter indlæses den aktuelle sample's species-liste ind i ArrayListen species. Da en species skal vide hvilken sample den tilhører bliver jeg nødt til at kalde til databasen for at få det højeste id, dvs. id'et på den sample jeg lige har sat ind.
Med species-listen og sample-id'et gås nu videre til species-løkken der fungerer på samme måde og ligeledes eksekveres Ab-løkken.
Hvis alt er gået efter planen er datafilen nu indlæst, og data ligger som det skal i databasen.
Hvis det er en Excel-fil der skal indlæses fungerer det på samme måde, bortset fra at der her bruges kolonne-hashtabellen til at fortælle hvor hvilke værdier er.

Dataudtrækning

Når brugeren har valgt søgekriterier og filnavn til output-filen på brugergrænsefladen, under fanebladet "Udtræk Data til EARSS", der kan ses i afsnittet om brugergrænsefladen, trykkes på knappen "Udtræk Data".
Når dette sker sendes søgekriterierne og filnavnet, via *PL_Facade*, til funktionen *extractEarssData* i en instans af klassen *ExtractionEngine*, der, som navnet antyder, styrer udtrækket af data. *extractEarssData* opretter en instans af klassen *QueryBuilder* sender søgekriterierne til den og får en forespørgsel retur. Forespørgslen sendes via *LD_Facade* til et objekt af klassen *DB_Connection*. *DB_Connection* opretter derpå forbindelse til databasen og kalder en Stored Procedure, *getEarssData*, på databasen, der returnerer det matchende data, hvis der er noget, i en *NpgsqlDataReader*. En *NpgsqlDataReader* er det objekt man bruger til at læse resultatet af en

forespørgsel til en PostgreSQL database ind i. Data bliver derefter returneret hele vejen tilbage til *ExtractionEngine*, der sender det videre til funktionen *buildEarssString* i en instans af klassen *DataStringBuilder*. *buildEarssString* sammensætter alle de returnerede værdier, i en StringBuilder. Det bliver gjort så det passer med EARSS-formatet, der kræver at værdierne står i en bestemt rækkefølge og adskilles med en Tab. Indtil videre indsendes kun de obligatoriske værdier til EARSS, derfor indsættes kun en Tab der hvor de ellers skulle have været.
Når data er samlet i StringBuilder'en returneres det i en string til *ExtractionEngine*. *ExtractionEngine* sender derefter tekststrengen og filnavnet, der blev angivet af brugeren, via *LD_Facade*, til en instans af *TextFileWriter* klassen, der udskriver den i en almindelig .txt-fil.

ErrorHandling

ErrorHandling burde nok have heddet noget i retning af ErrorLogging, da den som sådan ikke håndterer fejlene der opstår i programmet.
I stedet skriver den en fejllog ud i en tekstfil med de fejl der måtte opstå når programmet kører.
Jeg har valgt at lave klassen *ErrorHandler* statisk. Det synes jeg er den bedste måde, når jeg skal lave en fejllog som principielt alle andre klasser skal have mulighed for at skrive i. På denne måde behøver de forskellige klasser bare at importere *ErrorHandling*-pakken, og kalde *ErrorHandler.AddToErrorLog* for at skrive i fejlloggen.

```
try{
    reader = new XmlTextReader(filePath);
}
catch{
    ErrorHandler.AddToErrorLog("XmlFileReader.getXmlTextReader: " +
                               "Error loading xml file: " + filePath);
}
```

Jeg synes det er en bedre løsning end at lade hver klasse oprette sit eget objekt af klassen, der så alligevel skal skrive til den samme fejllog. *ErrorHandler* klassen indeholder en StringBuilder og to metoder, *AddToErrorLog* og *saveErrorLog*. *AddToErrorLog* metoden tager en string som parameter, som den bare ligger til StringBuilder'en.
saveErrorLog gemmer, som navnet antyder, loggen i en fil ved at sende fejlteksten til TextFileWriter klassen, i datalaget. Som den eneste klasse kalder ErrorHandler ikke LD_Facade, da den ikke hører til i noget bestemt lag, men kendes af alle klasser.

Test

Imens jeg har skrevet programmet har jeg naturligvis testet de enkelte dele undervejs som de er blevet færdige. Efter jeg blev færdig har jeg lavet større tests for at se om det hele nu også opførte sig som det skulle.
Her viste det sig at der var en fejl i ADBaktReader der gjorde at programmet nogle gange gik ned når det skulle oversætte Excel-filer.
Fejlen var forholdsvis omfattende, så jeg besluttede mig for at skrive selve Excel-parseren om, men kunne ikke rigtigt få det til at virke ordentligt. Til sidst har jeg valgt at lade det ligge, og sørge for at få gjort rapporten ordentlig færdig.
Jeg kan køre udtræksdelen, og den udskriver EARSS-data efter søgekriterierne i en tekst-fil og virker som den skal. Indlæsningsdelen virker som sagt ikke i øjeblikket for ADBaktReader's vedkommende, men Mads-delen bør, virke. Dette vil dog først vise sig når jeg har lavet ADBaktReader ordentligt, og kan teste det hele sammen igen, så jeg er sikker på at de ændringer jeg laver, ikke påvirker Mads-delen. Jeg vil dog mene at selve strukturen i ADBaktReader forbliver som den er beskrevet i design- og implementeringsafsnittet.

Udvidelser

Som det ser ud nu skal jeg fortsætte med at arbejde i Afdelingen for Antibiotikaresistens og Sygehushygiejne efter jeg har færdiggjort dette projekt.
Jeg har derfor gjort mig nogle overvejelser vedrørende udvidelser til programmet, der kunne gøre det endnu mere nyttigt i forbindelse med overvågningen af antibiotikaresistens.

Statistikmodul.

Jeg har bl.a. tænkt på at lave et statistikmodul. I første omgang er databasen jo bare "mellemstation" for data inden det sendes videre til EARSS. Men efterhånden som der kommer mere og mere data, vil det være nyttigt at kunne lave statistik på databasen, og se om der er nogen tendenser over tid. Da meget af resistensovervågningen går ud på at fremvise resultaterne i enten artikler, rapporter eller på hjemmesider, vil det være nyttigt hvis modulet kan tegne statistiske grafer og diagrammer, så det bliver let at præsentere.

Rapportmodul

Det kunne også blive nyttigt med et rapportmodul. Med et rapportmodul vil man kunne opsummere og udskrive oplysninger om antallet af rapporterede patientprøver, andelen af resistente prøver osv. for de enkelte KMA'er, regioner eller hele landet. Disse vil f.eks. kunne sendes til de enkelte KMA'er som feedback på de data de indsender.

Eksportmodul

Da jeg var i praktik i afdelingen startede jeg på at lave en hjemmeside til en gruppe der hedder DANRES-gruppen. DANRES-gruppen består af repræsentanter fra landet KMA'er, SSI og andre der har interesse i

antibiotikaresistens i Danmark. Hjemmesiden skal med tiden bl.a. fremvise resistens resultater statistik osv. over bl.a. de data der sendes til EARSS. Det vil derfor være oplagt at lave et modul til programmet der kan eksportere data eller statistikresultater i forskellige formater til brug for hjemmesiden, f.eks. HTML, XML eller lignende.

Konklusion

Produktet

Jeg har i dette projekt lavet et program til hjælp for indrapportering af resistensdata til EARSS. Desuden har jeg oprettet en database der kan indeholde disse data og på sigt vil kunne hjælpe til at give et godt overblik over antibiotikaresistente bakteriers udvikling i Danmark.
Testen har vist at der var en alvorlig fejl i ADBaktReader-klassen som jeg har ikke har fået rettet, pga. tidspres. Det vil sige at denne del af programmet ikke virker lige nu, men forhåbentlig kommer til det indenfor de nærmeste dage.
Dataudtræksdelen og databasen fungerer som de skal, og som jeg havde planlagt.

Processen

Jeg synes at arbejdsprocessen er gået forholdsvis godt. Jeg har stort set holdt mig til min tidsplan, selvom sådan en jo pr. definition altid er for optimistisk. Jeg brugte muligvis lidt for meget tid på at analysere og designe i starten, og kom lidt sent i gang med at programmere, men alt i alt synes jeg det var i orden.
Den store forskel mellem dette projekt og andre tilsvarende jeg har lavet er at jeg har været alene.
At lave projektet alene frem for i en gruppe har både haft fordele og ulemper. En klar fordel har været at jeg ved at have lavet det hele selv har haft helt styr på hvad der foregår hvor i projektet, hvilket sjældent er tilfældet når man arbejder i gruppe.
På det mere egoistiske plan har det også været rart at kunne lave tingene præcis som man ønskede det, og ikke har været nødt til at gå på kompromis, som det kan være tilfældet i en gruppe, hvis man ikke er enige om hvordan tingene skal laves. Modsat kan det være rart at høres andres idéer om

hvordan det kan gøres hvilket jeg har savnet lidt, da jeg næppe finder på den smarteste løsning hver gang.
En ulempe har dog været at man, af naturlige årsager, ikke når lige så meget når man er alene som når man er i en gruppe på 2-3 personer. Selv om jeg har lavet flere projekter alene, om end i en mindre skala, har det overrasket mig lidt at tingene tog så lang tid som det har gjort. Jeg havde egentlig regnet med at jeg ville nå mere og lige have en ekstra uge i reserve.

Samarbejdet med afdelingen

Samarbejdet med afdelingen har fungeret rigtigt godt.
Jeg har altid fået den hjælp jeg har skullet bruge, hvilket ikke har været så lidt, når man kommer ind i en ukendt verden, som mikrobiologi har været for mig.
Hvis alt går efter planen, bliver jeg ansat i afdelingen til bl.a. at udvide programmet samt at fortsætte indsamlingen og konverteringen af resistensdata, når jeg er færdig med uddannelsen.

Tak

Til sidst vil jeg gerne sige tak til min DTU vejleder Ole Remmer for vejledning vedr. det tekniske og opgaveskrivningen, til min SSI vejleder Dominique L. Monnet for vejledning vedr. den mikrobiologiske del af projektet og endelig til resten af Afdelingen for Antibiotikaresistens og Sygehushygiejne, specielt Arno Muller, programmør i afdelingen, for gode råd og hjælp til diverse programmerings- og database problemer.

Bilag

Indhold på CD'en

- Rapporten
- Kildekoden i .cs-filer.
- Eksempler på hele KMA oversættelsesfiler.

Data Exchange Format

EARSS dataudvekslingsformat taget fra EARSS manualen 2005.

14. Data exchange format

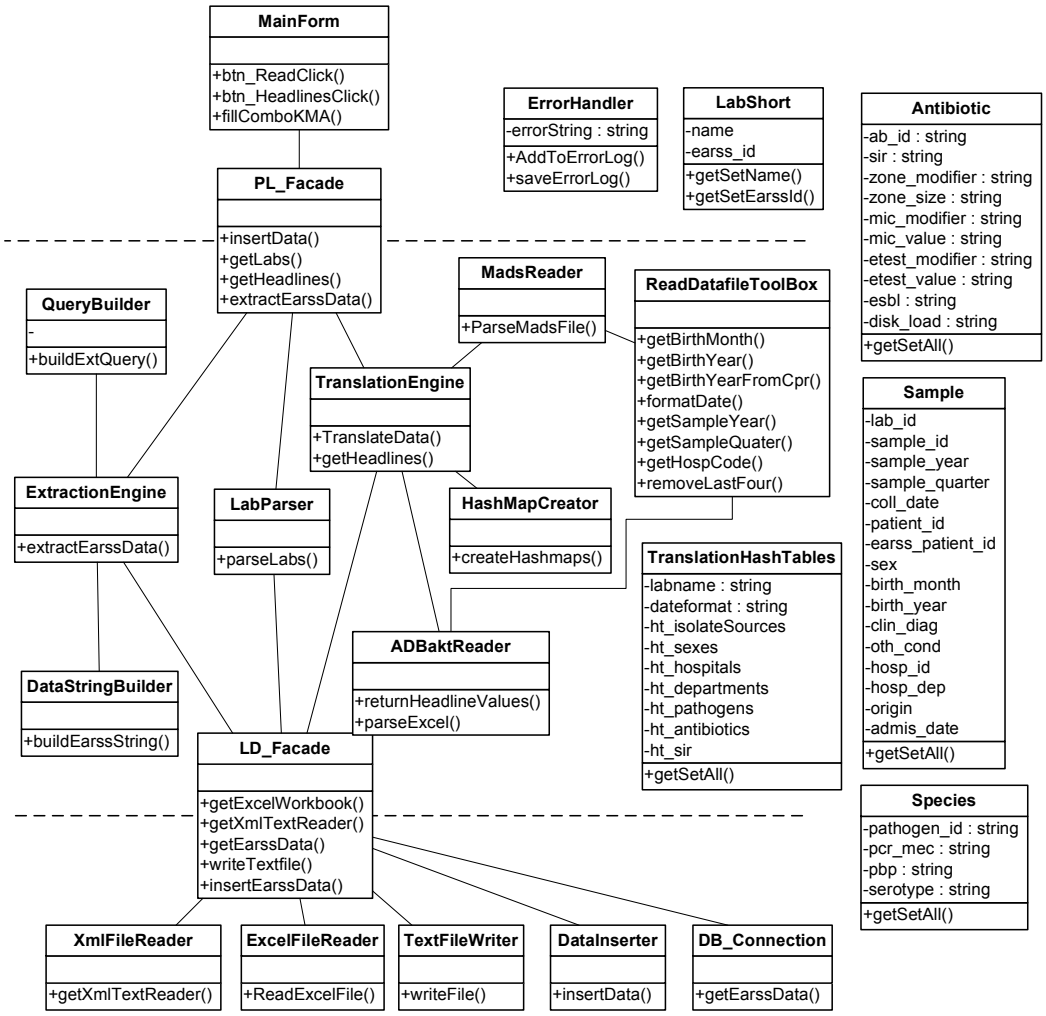
Brief description of data exchange format

Table 14.1. Brief description of data exchange format

Field Name	Field type	Initial column	Field length	Mandatory /Required /Optional
Current date	D	1	8	O
Laboratory code	A	9	5	M
Isolate sample number	A	14	12	R
Isolate source	A	26	2	R
Date of sample collection	D	28	8	M
Patient ID / Code	A	36	12	M
Sex	N	48	1	R
Month of birth	N	49	2	M
Year of birth	N	51	4	M
Clinical diagnosis	A	55	3	O
Other conditions	A	58	3	O
Hospital code	A	61	4	R
Origin of patient	N	65	1	R
Hospital department	A	66	3	R
Pathogen code	A	69	3	M
PCR mec-gene	N	72	1	O
Antibiotic code	A	73	3	M
S/I/R	A	76	1	M
Zone (> < =)	A	77	2	O
Zone (Value in mm)	A	79	2	O
MIC (> < =)	A	81	2	O
MIC (Value in mg/l)	A	83	5	O
E-test (> < =)	A	88	2	O
E-test (Value in mg/l)	A	90	5	O
Date of admission	D	95	8	O
ESBL present	N	103	1	O
Disk load	N	104	12	O
PBP2a-agglutination	N	116	1	O
Serotype	A	117	4	O

Field Name: Name of the variable, **Field type**: D= Date field, N=Numeric field, A=Alphabetic field, **Initial column**: First position of data field in case a fixed format is being used, **Field Length**: The field length is the total number of characters allowed for that field, **Mandatory**: Mandatory information to be included in the EARSS database, **Required**: Required to fit the EARSS protocols , **Optional**: Optional information is interesting to collect but not always available.

Klassediagram



Forkortet XML-fil til oversættelse

Forkortet oversættelsesfil til data fra Skejby Sygehus. Originalen er knap 5 gange så lang:

```
<?xml version="1.0" encoding="utf-8"?>
<data>
    <lab_id>DK014</lab_id>
    <date_format>YYYYMMDD</date_format>
    <isolate_sources>
        <source>
            <lab_code>Blod</lab_code>
             <earss_code>bl</earss_code>
        </source>
        <source>
            <lab_code>Spinalvæske</lab_code>
             <earss_code>sf</earss_code>
        </source>
        <alt_source>xx</alt_source>
    </isolate_sources>
    <sexes>
        <sex>
            <lab_code>M</lab_code>
            <earss_code>1</earss_code>
        </sex>
        <sex>
            <lab_code>F</lab_code>
            <earss_code>2</earss_code>
        </sex>
        <sex>
            <lab_code>K</lab_code>
            <earss_code>2</earss_code>
        </sex>
        <alt_sex>9</alt_sex>
    </sexes>
    <hospitals>
        <hospital>
            <lab_code>7002</lab_code>
            <earss_code>014A</earss_code>
        </hospital>
        <hospital>
            <lab_code>7003</lab_code>
            <earss_code>014B</earss_code>
        </hospital>
    </hospitals>
    <hospital_departments>
        <department>
            <dep_code>00</dep_code>
            <earss_dep>oth</earss_dep>
            <origin></origin>
        </department>
        <department>
            <dep_code>01</dep_code>
            <earss_dep>med</earss_dep>
```

```
                <origin></origin>
            </department>
            <department>
                <dep_code>04</dep_code>
                <earss_dep>hao</earss_dep>
                <origin></origin>
            </department>
            <department>
                <dep_code>30</dep_code>
                <earss_dep>sur</earss_dep>
                <origin></origin>
            </department>
            <alt_dep>xxx</alt_dep>
            <alt_orig>9</alt_orig>
        </hospital_departments>
        <antibiotic_codes>
             <ab>
                <lab_code>7</lab_code>
                <earss_code>ATM</earss_code>
            </ab>
            <ab>
                <lab_code>15</lab_code>
                <earss_code>CTX</earss_code>
            </ab>
            <ab>
                <lab_code>333</lab_code>
                earss_code>AMP</earss_code>
            </ab>
            <ab>
                <lab_code>358</lab_code>
                <earss_code>GEN</earss_code>
            </ab>
        </antibiotic_codes>
        <resistance>
            <sir>
                <lab_code>0</lab_code>
                <earss_code>R</earss_code>
            </sir>
             <sir>
                <lab_code>1</lab_code>
                <earss_code>I</earss_code>
            </sir>
             <sir>
                <lab_code>2</lab_code>
                <earss_code>S</earss_code>
            </sir>
        </resistance>
</data>
```

Kildekode

```
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;

namespace EarssDataConversion.DataClasses
{
	/// <summary>
	/// Description of Antibiotic.
	/// </summary>
	public class Antibiotic
	{
		string ab_id;
		string sir;
		string zone_modifier;
		string zone_size;
		string mic_modifier;
		string mic_value;
		string etest_modifier;
		string etest_value;
		string esbl;
		string disk_load;

		public Antibiotic()
		{
		}

		public string getSetAbId{
			get{return this.ab_id;}
			set{this.ab_id = value;}
		}
```

```
public string getSetSir{
      get{return this.sir;}
      set{this.sir = value;}
}

public string getSetZoneMod{
      get{return this.zone_modifier;}
      set{this.zone_modifier = value;}
}

public string getSetZoneSize{
      get{return this.zone_size;}
      set{this.zone_size = value;}
}

public string getSetMicMod{
      get{return this.mic_modifier;}
      set{this.mic_modifier = value;}
}

public string getSetMicValue{
      get{return this.mic_value;}
      set{this.mic_value = value;}
}

public string getSetEtestMod{
      get{return this.etest_modifier;}
      set{this.etest_modifier = value;}
}

public string getSetEtestValue{
      get{return this.etest_value;}
      set{this.etest_value = value;}
}

public string getSetEsbl{
      get{return this.esbl;}
      set{this.esbl = value;}
```

```
            }

            public string getSetDiskLoad{
                  get{return this.disk_load;}
                  set{this.disk_load = value;}
            }
      }
}
// End of Antibiotic
-------------------------------------------------------------------------------

/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;

namespace EarssDataConversion.DataClasses
{
      /// <summary>
      /// Description of LabShort.
      /// </summary>
      public class LabShort
      {
            string name;
            string earss_id;
            public LabShort()
            {
            }

            public string getSetName{
                  get{return this.name;}
                  set{this.name = value;}
            }

            public string getSetEarssId{
```

```
                get{return this.earss_id;}
                set{this.earss_id = value;}
            }
      }
}
// End of LabShort
-------------------------------------------------------------------------------

/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;

namespace EarssDataConversion.DataClasses
{
      /// <summary>
      /// Description of Sample.
      /// </summary>
      public class Sample
      {
            string lab_id;
            string sample_id;
            string sample_year;
            string sample_quarter;
            string source;
            string coll_date;
            string patient_id;
            string earss_patient_id;
            string sex;
            string birth_month;
            string birth_year;
            string clin_diag;
            string oth_cond;
            string hosp_id;
```

```
string hosp_dep;
string origin;
string admis_date;
ArrayList species;

public Sample()
{
}

public string getSetLabId{
      get{return this.lab_id;}
      set{this.lab_id = value;}
}

public string getSetSampleId{
      get{return this.sample_id;}
      set{this.sample_id = value;}
}

public string getSetSampleYear{
      get{return this.sample_year;}
      set{this.sample_year = value;}
}

public string getSetSampleQuarter{
      get{return this.sample_quarter;}
      set{this.sample_quarter = value;}
}

public string getSetSource{
      get{return this.source;}
      set{this.source = value;}
}

public string getSetCollDate{
      get{return this.coll_date;}
      set{this.coll_date = value;}
}
```

```
public string getSetPatientId{
      get{return this.patient_id;}
      set{this.patient_id = value;}
}

public string getSetEarssPatientId{
      get{return this.earss_patient_id;}
      set{this.earss_patient_id = value;}
}

public string getSetSex{
      get{return this.sex;}
      set{this.sex = value;}
}

public string getSetBirthMonth{
      get{return this.birth_month;}
      set{this.birth_month = value;}
}

public string getSetBirthYear{
      get{return this.birth_year;}
      set{this.birth_year = value;}
}

public string getSetClinDiag{
      get{return this.clin_diag;}
      set{this.clin_diag = value;}
}

public string getSetOthCond{
      get{return this.oth_cond;}
      set{this.oth_cond = value;}
}
public string getSetHospId{
      get{return this.hosp_id;}
      set{this.hosp_id = value;}
```

```
            }
            public string getSetHospDep{
                  get{return this.hosp_dep;}
                  set{this.hosp_dep = value;}
            }
            public string getSetOrigin{
                  get{return this.origin;}
                  set{this.origin = value;}
            }
            public string getSetAdmisDate{
                  get{return this.admis_date;}
                  set{this.admis_date = value;}
            }

            public void addSpecies(Object species){
                  this.species.Add(species);
            }

            public ArrayList getSpecies(){
                  return this.species;
            }
      }
}
// End of Sample
------------------------------------------------------------------------------

/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;

namespace EarssDataConversion.DataClasses
{
      /// <summary>
```

```
/// Description of Species.
/// </summary>
public class Species
{
	string pathogen_id;
	string pcr_mec;
	string pbp;
	string serotype;
	ArrayList antibiotics;

	public Species()
	{
	}

	public string getSetPathId{
		get{return this.pathogen_id;}
		set{this.pathogen_id = value;}
	}

	public string getSetPcrMec{
		get{return this.pcr_mec;}
		set{this.pcr_mec = value;}
	}

	public string getSetPbp{
		get{return this.pbp;}
		set{this.pbp = value;}
	}

	public string getSetSerotype{
		get{return this.serotype;}
		set{this.serotype = value;}
	}

	public void AddAntibiotic(Object ab){
		this.antibiotics.Add(ab);
	}
```

```
            public ArrayList getAntibiotics(){
                  return this.antibiotics;
            }
      }
}
// End of Species
-------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;

namespace EarssDataConversion.DataClasses
{
      /// <summary>
      /// Description of TranslationHashMaps.
      /// </summary>
      public class TranslationHashTables
      {
            string labname;
            string dateformat;

            Hashtable ht_isolateSources = null;
            Hashtable ht_sexes = null;
            Hashtable ht_hospitals = null;
            Hashtable ht_departments = null;
            Hashtable ht_pathogens = null;
            Hashtable ht_antibiotics = null;
            Hashtable ht_sir = null;

            public void TranslationHashMaps()
            {
                  ht_isolateSources = new Hashtable();
                  ht_sexes  = new Hashtable();
```

```
        ht_hospitals  = new Hashtable();
        ht_departments  = new Hashtable();
        ht_pathogens  = new Hashtable();
        ht_antibiotics  = new Hashtable();
        ht_sir = new Hashtable();
}

public string getSetLabname{
        get{return this.labname;}
        set{this.labname = value;}
}

public string getSetDateformat{
        get{return this.dateformat;}
        set{this.dateformat = value;}
}

public void addIsoSource(string key, string earss_value){
        ht_isolateSources.Add(key, earss_value);
}

public string getIsoSource(string key){
        return (string)ht_isolateSources[key];
}

public void addSex(string key, string earss_value){
        ht_sexes.Add(key, earss_value);
}

public string getSex(string key){
        return (string)ht_sexes[key];
}

public void addHospital(string key, string earss_value){
        ht_hospitals.Add(key, earss_value);
}

public string getHospital(string key){
```

```
        return (string)ht_hospitals[key];
}

public void addDepartment(string key, string earss_dep, string origin){
        string[] values = new string[2];
        values[0] = earss_dep;
        values[1] = origin;

        ht_departments.Add(key, values);
}

public string[] getDepartment(string key){
        return (string[])ht_departments[key];
}

public void addPathogen(string key, string earss_value){
        ht_pathogens.Add(key, earss_value);
}

public string getPathogen(string key){
        return (string)ht_pathogens[key];
}

public void addAntibiotic(string key, string earss_value){
        ht_antibiotics.Add(key, earss_value);
}

public string getAntibiotic(string key){
        return (string)ht_antibiotics[key];
}

public void addSir(string key, string earss_value){
        ht_sir.Add(key, earss_value);
}

public string getSir(string key){
        return (string)ht_sir[key];
}
```

```
	}
}
// End of TranslationHashTables


------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Data;
using System.Collections;
using Npgsql;
using EarssDataConversion.DataClasses;
using EarssDataConversion.ErrorHandling;


namespace EarssDataConversion.DataConnections
{
	/// <summary>
	/// Description of DataInserter.
	/// </summary>
	public class DataInserter
	{
		public DataInserter()
		{
		}

		public void insertData(ArrayList translatedData){
			NpgsqlConnection conn = new NpgsqlConnection("Server=127.0.0.1;Port=5432;User
ID=postgres;password=1234;Database=postgres;");
			NpgsqlDataReader rdr = null;
			Hashtable abIDs = new Hashtable();
			Hashtable labs = new Hashtable();
```

```
                NpgsqlCommand getAb = new NpgsqlCommand("SELECT id, earss_name FROM tb_antibiotics;",conn);

                NpgsqlCommand getLabs = new NpgsqlCommand("SELECT id, name FROM tb_labs;",conn);


                NpgsqlCommand insertSample = new NpgsqlCommand("INSERT INTO tb_samples(" +
                                                               "sample_id_lab, sample_year, " +
                                                               "sample_quarter, id_lab, sample_source, " +
                                                               "coll_date, patient_seq_number, " +
                                                               "earss_patient_seq_number, birthyear, " +
                                                               "birthmonth, sex, hosp_code, hosp_department,
origin) " +
                                                               "VALUES(:sid, :syear, :squarter, :idlab,
:ssource, :cdate," +
                                                               ":pseqnum, :epseqnum, :byear, :bmonth, :sex,
:hcode, :hdep, :orig) ",conn);
                NpgsqlCommand insertSpecies = new NpgsqlCommand("INSERT INTO tb_species(id_pathogen,
id_sample) " +
                                                                "VALUES( :pathogen,:sampleid)",conn);
                NpgsqlCommand insertAb = new NpgsqlCommand("INSERT INTO tb_ab(id_antibiotic, id_species, sir)
" +
                                                           "VALUES(:abid, :specid, :sir)",conn);
                NpgsqlCommand getMaxSampleID = new NpgsqlCommand("SELECT Max(id) FROM tb_samples",conn);
                NpgsqlCommand getMaxSpeciesID = new NpgsqlCommand("SELECT Max(id) FROM tb_species",conn);

                ArrayList speciesList = null;
                ArrayList abList = null;
                string sampleMax;
                string speciesMax;
                int intSampleMax = 0;
                int intSpeciesMax = 0;

                if(translatedData != null){

                      conn.Open();

                      //Get antibiotic id from antibiotic names and put them in a hashtable
                      try{
```

```
      rdr = getAb.ExecuteReader();
}
catch{ErrorHandler.AddToErrorLog("DataInserter.insertData: Error getting ab id's");}

if(rdr != null){
      while(rdr.Read()){
            abIDs.Add(rdr[1].ToString(), rdr[1].ToString());
      }
}
//Get labs id from lab names and put them in a hashtable
try{
      rdr = getLabs.ExecuteReader();
}
catch{ErrorHandler.AddToErrorLog("DataInserter.insertData: Error getting labs");}

if(rdr != null){
      while(rdr.Read()){
            labs.Add(rdr[1].ToString(), rdr[1].ToString());
      }
}
```

```
insertSample.Parameters.Add(new NpgsqlParameter(":sid", DbType.String ));
insertSample.Parameters.Add(new NpgsqlParameter(":syear", DbType.Int16));
insertSample.Parameters.Add(new NpgsqlParameter(":squarter", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":idlab", DbType.Int32));
insertSample.Parameters.Add(new NpgsqlParameter(":ssource", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":cdate", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":pseqnum", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":epseqnum", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":byear", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":bmonth", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":sex", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":hcode", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":hdep", DbType.String));
insertSample.Parameters.Add(new NpgsqlParameter(":orig",DbType.Int16));
```

```
insertSpecies.Parameters.Add(new NpgsqlParameter(":pathogen", DbType.String));
insertSpecies.Parameters.Add(new NpgsqlParameter(":sampleid", DbType.Int32));


insertAb.Parameters.Add(new NpgsqlParameter(":abid", DbType.Int32));
insertAb.Parameters.Add(new NpgsqlParameter(":specid", DbType.Int32));
insertAb.Parameters.Add(new NpgsqlParameter(":sir", DbType.String));

insertSample.Prepare();
insertSpecies.Prepare();
insertAb.Prepare();

foreach(Sample sa in translatedData){
      insertSample.Parameters[0].Value = sa.getSetSampleId;
      insertSample.Parameters[1].Value = sa.getSetSampleYear;
      insertSample.Parameters[2].Value = sa.getSetSampleQuarter;
      insertSample.Parameters[3].Value = labs[sa.getSetLabId];
      insertSample.Parameters[4].Value = sa.getSetSource;
      insertSample.Parameters[5].Value = sa.getSetCollDate;
      insertSample.Parameters[6].Value = sa.getSetPatientId;
      insertSample.Parameters[7].Value = sa.getSetEarssPatientId;
      insertSample.Parameters[8].Value = sa.getSetBirthYear;
      insertSample.Parameters[9].Value = sa.getSetBirthMonth;
      insertSample.Parameters[10].Value = sa.getSetSex;
      insertSample.Parameters[11].Value = sa.getSetHospId;
      insertSample.Parameters[12].Value = sa.getSetHospDep;
      insertSample.Parameters[13].Value = sa.getSetOrigin;

      try{insertSample.ExecuteReader();}
      catch{ErrorHandler.AddToErrorLog("DataInserter.insertData: Error inserting
sample");}

      speciesList = sa.getSpecies();
      try{
            rdr = getMaxSampleID.ExecuteReader();
            sampleMax = rdr[0].ToString();
            int.TryParse(sampleMax,out intSampleMax);
      }
```

```
                        catch{ErrorHandler.AddToErrorLog("DataInserter.insertData: Error getting
sampleMax");}

                        foreach(Species sp in speciesList){
                              insertSpecies.Parameters[0].Value = sp.getSetPathId;
                              insertSpecies.Parameters[1].Value = intSampleMax;

                              try{insertSpecies.ExecuteReader();}
                              catch{ErrorHandler.AddToErrorLog("DataInserter.insertData: Error inserting
species");}

                              try{
                                    rdr = getMaxSpeciesID.ExecuteReader();
                                    speciesMax = rdr[0].ToString();
                                    int.TryParse(speciesMax,out intSpeciesMax);
                              }
                              catch{ErrorHandler.AddToErrorLog("DataInserter.insertData: Error getting
speciesMax");}

                              abList = sp.getAntibiotics();
                              foreach(Antibiotic ab in abList){
                                    insertAb.Parameters[0].Value = abIDs[ab.getSetAbId];
                                    insertAb.Parameters[1].Value = intSpeciesMax;
                                    insertAb.Parameters[2].Value = ab.getSetSir;

                                    try{
                                          insertAb.ExecuteReader();
                                    }
                                    catch{ErrorHandler.AddToErrorLog("DataInserter.insertData: Error
inserting ab");}
                              }
                        }
                  }
                  conn.Close();
            }
            else
                  ErrorHandler.AddToErrorLog("DataInserter.insertData: translatedData = null");
      }
```

```
	}
}
// End of DataInserter

-----------------------------------------------------------------------------

/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Data;
using Npgsql;
using EarssDataConversion.ErrorHandling;

namespace EarssDataConversion.DataConnections
{
	/// <summary>
	/// Description of DB_Connection.
	/// </summary>
	public class DB_Connection
	{
		public DB_Connection()
		{

		}

		public NpgsqlDataReader getEarssData(string query){
			NpgsqlConnection conn = new NpgsqlConnection("Server=127.0.0.1;Port=5432;User
ID=postgres;password=1234;Database=postgres;");
			NpgsqlCommand comm = new NpgsqlCommand(query,conn);
			NpgsqlDataReader rdr = null;

			try
			{
				conn.Open();
```

```
                        rdr = comm.ExecuteReader();
                        conn.Close();
                  }
                  catch(Exception e)
                  {ErrorHandler.AddToErrorLog("DB_Connection.getEarssData: " +
                                              "Fejl i databaseforbindelse: "+e);}
                  return rdr;
            }
      }
}
// End of DB_Connection

-------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Data.OleDb;
using Excel;
using EarssDataConversion.ErrorHandling;

namespace EarssDataConversion.DataConnections
{
      /// <summary>
      /// Description of ExcelFileReader.
      /// </summary>
      public class ExcelFileReader
      {
            public ExcelFileReader()
            {
            }

            public Excel.Workbook ReadExcelFile (string filePath){
                  object missingValue = System.Reflection.Missing.Value;
                  Excel._Application eApp = new Application();
```

```
                Excel.Workbook wb = null;
                eApp.Visible = false;

                try{
                      wb = eApp.Workbooks.Open(filePath,0, true, 5, "", "", true,
                                               Excel.XlPlatform.xlWindows, "\t",false,
                                               false,0,true,true,true);

                }
                catch{ErrorHandler.AddToErrorLog("ExcelFileReader.ReadExcelFile: Error opening workbook");}

                return wb;
          }
     }
}
// End of ExcelFileReader


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.IO;

namespace EarssDataConversion.DataConnections
{
      /// <summary>
      /// Description of TextFileWriter.
      /// </summary>
      public class TextFileWriter
      {
            public TextFileWriter()
            {
            }
```

```
            public void writeFile(string filePath, string fileContent){
                  FileStream fs = new FileStream(filePath, FileMode.OpenOrCreate, FileAccess.Write);

                  StreamWriter sw = new StreamWriter(fs);

                  try{
                        sw.WriteLine(fileContent);
                  }
                  finally{
                        if(sw != null){sw.Close();}
                  }
            }
      }
}
// End of TextFileWriter


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Xml;
using EarssDataConversion.ErrorHandling;

namespace EarssDataConversion.DataConnections
{
      /// <summary>
      /// Description of Translation_Xml_Reader.
      /// </summary>
      public class XmlFileReader
      {
            public XmlFileReader()
            {
```

```
            }

            public XmlTextReader getXmlTextReader(string filePath){
                  XmlTextReader reader = null;

                  try{
                        reader = new XmlTextReader(filePath);
                  }
                  catch{
                        ErrorHandler.AddToErrorLog("XmlFileReader.getXmlTextReader: " +
                                                   "Error loading xml file: " + filePath);
                  }
                  return reader;
            }
      }
}
// End of XmlFileReader


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Text;
using System.Collections;
using Npgsql;
using EarssDataConversion.ErrorHandling;

namespace EarssDataConversion.DataExtraction
{
      /// <summary>
      /// Description of DataStringBuilder.
      /// </summary>
      public class DataStringBuilder
```

```
{
	public DataStringBuilder()
	{
	}

	public string buildEarssString(NpgsqlDataReader data){
		StringBuilder stb = new StringBuilder();

		if(data != null){

			while(data.Read()){
				if(data.FieldCount>13){
					stb.Append("\t");
					stb.Append(data[0].ToString() + "\t");
					stb.Append(data[1].ToString() + "\t");
					stb.Append(data[2].ToString() + "\t");
					stb.Append(data[3].ToString() + "\t");
					stb.Append(data[4].ToString() + "\t");
					stb.Append(data[5].ToString() + "\t");
					stb.Append(data[6].ToString() + "\t");
					stb.Append(data[7].ToString() + "\t");
					stb.Append("\t");
					stb.Append("\t");
					stb.Append(data[8].ToString() + "\t");
					stb.Append(data[9].ToString() + "\t");
					stb.Append(data[10].ToString() + "\t");
					stb.Append(data[11].ToString() + "\t");
					stb.Append("\t");
					stb.Append(data[12].ToString() + "\t");
					stb.Append(data[13].ToString() + "\t");
					stb.Append("\t");
					stb.Append("\t");
					stb.Append("\t");
					stb.Append("\t");
					stb.Append("\t");
					stb.Append("\t");
					stb.Append("\t");
					stb.Append("\t");
```

```
                                    stb.Append("\t");
                                    stb.Append("\t");
                                    stb.Append("\t");

                                    stb.Append("\r\n");
                              }
                        }
                        return stb.ToString();
                  }
                  else{
                        ErrorHandler.AddToErrorLog("DataStringBuilder.buildEarssString: NpgsqlDataReader =
null");

                        return "Der opstod en fejl";
                  }
            }
      }
}
// End of DataStringBuilder


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;
using EarssDataConversion.Facades;
using Npgsql;

namespace EarssDataConversion.DataExtraction
{
      /// <summary>
      /// Description of ExtractionEngine.
      /// </summary>
      public class ExtractionEngine
```

```
	{
		public ExtractionEngine()
		{
		}

		public void extractEarssData(string filePath, string year, string quarter, ArrayList pathogens){
			QueryBuilder qb = new QueryBuilder();
			LD_Facade ldf = new LD_Facade();
			NpgsqlDataReader earssData = null;
			DataStringBuilder dsb = new DataStringBuilder();

			earssData = ldf.getEarssData(qb.buildExtQuery(year, quarter, pathogens));
			ldf.writeTextFile(filePath, dsb.buildEarssString(earssData));

		}
	}
}
// End of ExtractionEngine


-------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;
using System.Text;

namespace EarssDataConversion.DataExtraction
{
	/// <summary>
	/// Description of QueryBuilder.
	/// </summary>
	public class QueryBuilder
	{
```

```
            public QueryBuilder()
            {
            }

            public string buildExtQuery(string year, string quarter, ArrayList pathogens){
                  StringBuilder species = new StringBuilder();
                  StringBuilder dbQuery = new StringBuilder();
                  species.Append("'IN (");
                  foreach (String s in pathogens){
                        species.Append("''" + s.ToString() + "',");
                  }
                  species.Remove(species.Length-1,1);
                  species.Append(")'");

                  dbQuery.Append( "select * from getearssdata('"+quarter+"','"+year+"',"+species+
                                "") AS (earss_id varchar, sample_id_lab varchar, sample_source varchar," +
                                " coll_date text, earss_patient_seq_number varchar, sex varchar, " +
                                "birthmonth text, birthyear text, hosp_code varchar, origin int2, " +
                                "hosp_department varchar, id_pathogen text, earss_name varchar, sir " +
                                "bpchar)");

                  return dbQuery.ToString();
            }

      }
}
// End of QueryBuilder


---------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;
```

```
using Excel;
using EarssDataConversion.ErrorHandling;
using EarssDataConversion.DataClasses;


namespace EarssDataConversion.DataInsertion.ReadDatafiles
{
	/// <summary>
	/// Description of ADBaktReader.
	/// </summary>
	public class ADBaktReader
	{
		private ArrayList headlineValues;

		Excel.Workbook wb = null;

		public ADBaktReader(Excel.Workbook wb)
		{
			this.wb = wb;
		}

		public ArrayList returnHeadlineValues(){
			if (wb != null){
				Excel.Worksheet ws = (Excel.Worksheet)wb.Worksheets[1];
				headlineValues = new ArrayList();
				Range usedCols = ws.UsedRange.Columns;

				for(int i=1; i< usedCols.Count ;i++){
					Console.WriteLine(i);
					headlineValues.Add(((Excel.Range)ws.Cells[1,i]).Value2);
				}
			}
			else{
				ErrorHandler.AddToErrorLog("ExcelReader.ReturnHeadlineValues: wb = null");
			}

			return headlineValues;
		}
```

```
            public ArrayList parseExcel(TranslationHashTables tht, Hashtable colTab, string labId){
                  ArrayList translatedData = null;
                  if (wb != null){
                        Excel.Worksheet ws = (Excel.Worksheet)wb.Worksheets[1];
                        Range r = (Range)ws.UsedRange.Columns;
                        ReadDatafileToolbox toolbox = new ReadDatafileToolbox();
                        string dateFormat = tht.getSetDateformat;
                        string[] OrigDep = new string[2];

                        for (int i=1; i < r.Count; i++){
                              if(translatedData.Count < 1 ){
                                    Sample sample = new Sample();
                                    sample.getSetLabId = labId;
                                    sample.getSetSampleId = (ws.Cells[i, colTab["sampleId"]].ToString());
                                    sample.getSetSource =
tht.getIsoSource((string)((Excel.Range)(ws.Cells[i,colTab["source"]])).Value2);
                                    sample.getSetCollDate =
toolbox.formatDate((string)((Excel.Range)(ws.Cells[i,colTab["collDate"]])).Value2,tht.getSetDateformat);
                                    sample.getSetPatientId = (ws.Cells[i,colTab["patientId"]].ToString());
                                    sample.getSetSex =
tht.getSex((string)((Excel.Range)(ws.Cells[i,colTab["sex"]])).Value2);
                                    sample.getSetBirthMonth = toolbox.getBirthMonth(((Excel.Range)ws.Cells[i,
colTab["birhtMonth"]]).Value2.ToString());
                                    sample.getSetBirthYear =
toolbox.getBirthYearFromCpr(((Excel.Range)ws.Cells[i,colTab["birthYear"]]).Value2.ToString());
                                    sample.getSetHospId =
tht.getHospital(toolbox.getHospCode((string)((Excel.Range)(ws.Cells[i,colTab["hospCode"]])).Value2));
                                    OrigDep =
(string[])tht.getDepartment((string)((Excel.Range)(ws.Cells[i,colTab["hospDep"]])).Value2);
                                    sample.getSetHospDep = OrigDep[0];
                                    sample.getSetOrigin = OrigDep[1];
                              }

                              //if( (Excel.Range)ws.Cells[i,colTab["sampleId"]]

 //********************IKKE FÆRDIG!!!*************************************************
                        }
```

```
                }
                else{
                        ErrorHandler.AddToErrorLog("ExcelReader.ParseExcel: wb = null");
                }

                return translatedData;
            }
      }
}
// End of ADBaktReader


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Xml;
using System.Collections;
using EarssDataConversion.DataClasses;

namespace EarssDataConversion.DataInsertion.ReadDatafiles
{
      /// <summary>
      /// Description of MadsReader.
      /// </summary>
      public class MadsReader
      {
            public MadsReader()
            {
            }

            public ArrayList ParseMadsFile(XmlTextReader xtr, TranslationHashTables tht, string labId){
                  ArrayList translatedData = new ArrayList();
                  Sample sample = null;
```

```
                Species species = null;
                Antibiotic ab = null;
                string dateFormat = tht.getSetDateformat;
                ReadDatafileToolbox rdt = new ReadDatafileToolbox();
                string birthday = "";
                string collDate = "";
                string patientId = "";

                while (xtr.Read()){
                      switch(xtr.NodeType){
                                  case XmlNodeType.Element:
                                        switch (xtr.Name){
                                              case "SAMPLEID":
                                                    //new sample
                                                    sample.getSetSampleId = xtr.ReadString();
                                                    sample.getSetLabId = labId;
                                                    break;

                                              case "REKV":
                                                    collDate = tht.getHospital(xtr.ReadString());
                                                    sample.getSetHospId = collDate;
                                                    sample.getSetSampleYear = rdt.getSampleYear(collDate);
                                                    sample.getSetSampleQuarter =
rdt.getSampleQuarter(collDate);
                                                    break;

                                              case "SAMPLETYPE":
                                                    sample.getSetSource = tht.getIsoSource(xtr.ReadString());
                                                    break;

                                              case "WARDTYPE":
                                                    string[] values = new string[2];
                                                    values = (string[])tht.getDepartment(xtr.ReadString());
                                                    sample.getSetHospDep = values[0];
                                                    sample.getSetOrigin = values[1];
                                                    break;
```

```
dateFormat);
```

```
rdt.removeLastFour(patientId);
```

```
case "RECVDATE":

      sample.getSetCollDate = rdt.formatDate(xtr.ReadString(),

      break;

case "SEQNUMBER":
      patientId = xtr.ReadString();
      sample.getSetPatientId = patientId;
      if(patientId[6] == '-')

            sample.getSetEarssPatientId =

      else
            sample.getSetEarssPatientId = patientId;
      break;

case "SEX":
      sample.getSetSex = tht.getSex(xtr.ReadString());
      break;

case "BIRTHDAY":

      birthday = xtr.ReadString();
      sample.getSetBirthMonth = rdt.getBirthYear(birthday);
      sample.getSetBirthYear = rdt.getBirthMonth(birthday);
      break;

case "TYPE":

      sample.getSetOrigin = xtr.ReadString();
      break;

case "SPECIESID":
      //new species

      species.getSetPathId = xtr.ReadString();
      break;
```

```
                        case "ABID":
                              //new ab

                              ab.getSetAbId = tht.getAntibiotic(xtr.ReadString());

                              break;

                        case "ABDIAG":

                              ab.getSetSir = tht.getSir(xtr.ReadString());
                              break;

                        case "ABZONE":

                              ab.getSetZoneSize = xtr.ReadString();
                              break;
                  }
                  break;
            case XmlNodeType.EndElement:
                  switch(xtr.Name){
                        case "AB":
                              species.AddAntibiotic(ab);
                              ab = null;
                              break;

                        case "species":
                              sample.addSpecies(species);
                              species = null;
                              break;

                        case "SAMPLE":
                              translatedData.Add(sample);
                              sample = null;
                              break;
                  }
                  break;
      }
```

```
                }
                return translatedData;
            }
        }
}
// End of MadsReader


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using EarssDataConversion.ErrorHandling;

namespace EarssDataConversion.DataInsertion.ReadDatafiles
{
      /// <summary>
      /// Description of ReadDatafileToolbox.
      /// </summary>
      public class ReadDatafileToolbox
      {
            public ReadDatafileToolbox()
            {
            }

            public string getBirthMonth(string birthday){
                  return birthday.Substring(2,2);
            }

            public string getBirthYear(string birthday){
                  int birthYear;
                  int.TryParse(birthday.Substring(4,2),out birthYear);

                  if (birthYear<6){
```

```
                        return "20" + birthYear;
                }
                else{
                        return "19" + birthYear;
                }
        }

        public string getBirthYearFromCpr(string cpr){

                string birthYear = "";
                if (cpr.Length == 11){

                        Console.WriteLine("I'm in");
                        int year;
                        int.TryParse(cpr.Substring(4,2), out year);
                        int lastFour;
                        int.TryParse(cpr.Substring(7,4), out lastFour);

                        if(lastFour<4000){
                                birthYear =  "19"+year;
                        }
                        else if(lastFour>=4000 && lastFour<5000){
                                if(year<37){
                                        return "20"+year;
                                }
                                else{
                                        birthYear =  "19"+year;
                                }
                        }
                        else if(lastFour>=5000 && lastFour<9000){
                                if(year<37){
                                        birthYear =  "20"+year;
                                }
                                else if(year>=37 && year<58){
                                        ErrorHandler.AddToErrorLog("ReadDatafileToolbox.getBirthYearFromCpr: not a
valid cpr");
                                        birthYear =  "xxxx";
                                }
```

```
                        else{
                              birthYear =  "18"+year;
                        }
                  }
                  else if(lastFour>=9000 && lastFour<10000){
                        if(year<37){
                              birthYear =  "20"+year;
                        }
                        else{
                              birthYear =  "19"+year;
                        }
                  }
                  else{
                        ErrorHandler.AddToErrorLog("ReadDatafileToolbox.getBirthYearFromCpr: Cpr not 11
characters");
                        birthYear =  "xxxx";
                  }
            }
            else
                  birthYear =  "xxxx";

            if(birthYear.Length == 3){
                  return birthYear.Substring(0,2)+"0"+birthYear.Substring(2,1);
            }
            else
                  return birthYear;
      }

      public string formatDate(string date, string format){
            string day;
            string month;
            string year;
            switch(format){
                  case "YYYYMMDD":
                        return (date);

                  case "DDMMYYYY":
                        day = date.Substring(0,2);
```

```
                  month = date.Substring(2,2);
                  year = date.Substring(4,4);
                  return ( year+month+day);

            case "MMDDYYYY":
                  day = date.Substring(2,2);
                  month = date.Substring(0,2);
                  year = date.Substring(4,4);
                  return (year+month+day);

            default:
                  ErrorHandler.AddToErrorLog("ReadDatafileToolbox.formatDate: unknown date format");
                  return date;
      }
}

public string getSampleYear(string inDate){
      return inDate.Substring(0,4);
}

public string getSampleQuarter(string inDate){
      int month;
      string quarter = "";
      int.TryParse(inDate.Substring(4,2), out month);

      if(month<4)
            quarter = "1";
      else
            if(month>3 && month<7)
                  quarter = "2";
            else
                  if(month>6 && month<10)
                        quarter ="3";
                  else
                        if(month>9 && month<13)
                              quarter ="4";
                        else
```

```
                                    ErrorHandler.AddToErrorLog("ReadDatafileToolbox.getSampleQuarter:
illegal month");

                  return quarter;
            }

            public string getHospCode(string longCode){
                  return longCode.Substring(0,2);
            }

            public string removeLastFour(string cpr){
                  return cpr.Substring(0,6);
            }
      }
}
// End of ReadDatafileToolBox


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;
using System.Xml;
using EarssDataConversion.DataClasses;
using EarssDataConversion.Facades;

namespace EarssDataConversion.DataInsertion.Translation
{
      /// <summary>
      /// Description of HashmapCreator.
      /// </summary>
      public class HashmapCreator
      {
```

```
public HashmapCreator()
{

}

public TranslationHashTables createHashmaps(string filePath){
      LD_Facade ldi = new LD_Facade();
      ArrayList hashTables = new ArrayList();
      XmlTextReader xtr = ldi.getXmlTextReader(filePath);
      string inTag = "none";
      string tmp_lab = "";
      string tmp_earss = "";

      TranslationHashTables tht = new TranslationHashTables();

      while (xtr.Read()){
            switch ( xtr.NodeType)
            {
                  case XmlNodeType.Element:
                        switch (xtr.Name)
                        {
                              case "date_format":
                                    tht.getSetDateformat = xtr.ReadString();
                                    break;

                              case "lab_id":
                                    tht.getSetLabname = xtr.ReadString();
                                    break;

                              case "isolate_sources":
                                    inTag = "iso";
                                    break;

                              case "sexes":
                                    inTag = "sex";
                                    break;

                              case "hospitals":
```

```
            inTag = "hosp";
            break;

      case "pathogens":
            inTag = "path";
            break;

      case "antibiotics":
            inTag = "ab";
            break;

      case "resistance":
            inTag = "sir";
            break;

      case "lab_code":

            switch (inTag)
            {
                  case "iso":
                        tmp_lab = xtr.ReadString();

                        xtr.Read();xtr.Read();xtr.Read();
                        tmp_earss = xtr.ReadString();

                        tht.addIsoSource(tmp_lab, tmp_earss);

                        break;

                  case "sex":
                        tmp_lab = xtr.ReadString();

                        xtr.Read();xtr.Read();xtr.Read();
                        tmp_earss = xtr.ReadString();

                        tht.addSex(tmp_lab, tmp_earss);

                        break;
```

```
case "hosp":
      tmp_lab = xtr.ReadString();

      xtr.Read();xtr.Read();xtr.Read();
      tmp_earss = xtr.ReadString();

      tht.addHospital(tmp_lab, tmp_earss);

      break;

case "path":
      tmp_lab = xtr.ReadString();

      xtr.Read();xtr.Read();xtr.Read();
      tmp_earss = xtr.ReadString();

      tht.addPathogen(tmp_lab, tmp_earss);

      break;

case "ab":
      tmp_lab = xtr.ReadString();

      xtr.Read();xtr.Read();xtr.Read();
      tmp_earss = xtr.ReadString();

      tht.addAntibiotic(tmp_lab, tmp_earss);

      break;

case "sir":
      tmp_lab = xtr.ReadString();
      xtr.Read();xtr.Read();xtr.Read();
      tmp_earss = xtr.ReadString();

      tht.addSir(tmp_lab, tmp_earss);
      break;
```

```
                    }
                    break;
              case "dep_code":
                    xtr.Read();
                    string tmp_dep = xtr.Value.ToString();
                    xtr.Read();xtr.Read();xtr.Read();xtr.Read();
                    string tmp_earss_dep = xtr.Value.ToString();
                    xtr.Read();xtr.Read();xtr.Read();xtr.Read();
                    string tmp_origin = xtr.Value.ToString();

                    tht.addDepartment(tmp_dep, tmp_earss, tmp_origin);
                    break;
        }
        break;
case XmlNodeType.EndElement:
        switch (xtr.Name)
        {
              case "isolate_source":
                    inTag = "none";
                    break;

              case "sexes":
                    inTag = "none";
                    break;

              case "hospitals":
                    inTag = "none";
                    break;

              case "pathogens":
                    inTag = "none";
                    break;

              case "antibiotics":
                    inTag = "none";
                    break;
        }
        break;
```

```
                        }
                    }
                    return tht;
                }
            }
}
// End of HashMapCreator


-----------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Xml;
using System.Collections;
using EarssDataConversion.Facades;
using EarssDataConversion.ErrorHandling;
using EarssDataConversion.DataClasses;
using EarssDataConversion.DataInsertion.ReadDatafiles;


namespace EarssDataConversion.DataInsertion.Translation
{
      /// <summary>
      /// Description of Translator.
      /// </summary>
      public class TranslationEngine
      {
            PL_Facade plf = null;
            LD_Facade ldf = null;
            MadsReader mr = null;
            XmlTextReader xtr = null;
            TranslationHashTables tht = null;
            ArrayList translatedData = null;
```

```
public TranslationEngine()
{
      plf = new PL_Facade();
      ldf = new LD_Facade();
}

public void TranslateData(Hashtable columnTable, string filePath, string labId){
      string fileType ="";

      if (filePath == "" || labId == "" ){
            ErrorHandler.AddToErrorLog("TranslationEngine: fileName or labName not set");
      }
      else{
            switch (labId)
            {
                  case "Rigshospitalet":
                        fileType = "xml";
                        break;

                  case "Hvidovre":
                        fileType = "xls";
                        break;

                  case "Herlev":
                        fileType = "xls";
                        break;

                  case "Hillerød":
                        fileType = "xls";
                        break;

                  case "Roskilde":
                        fileType = "xls";
                        break;

                  case "Slagelse":
                        fileType = "xls";
```

```
            break;

        case "Næstved":
            fileType = "xml";
            break;

        case "Odense":
            fileType = "xml";
            break;

        case "Sønderborg":
            fileType = "xml";
            break;

        case "Esbjerg":
            fileType = "xml";
            break;

        case "Vejle":
            fileType = "xml";
            break;

        case "Herning":
            fileType = "xml";
            break;

        case "Skejby":
            fileType = "xml";
            break;

        case "Viborg":
            fileType = "xml";
            break;

        case "Aalborg":
            fileType = "xls";
            break;
    }
```

```
                        if(fileType == "xml" || fileType == "xls"){
                              switch (fileType){
                                    case "xml":
                                          mr = new MadsReader();
                                          ldf = new LD_Facade();
                                          xtr = ldf.getXmlTextReader(filePath);

                                          tht = (new HashmapCreator().createHashmaps("../TranslationFiles/" +
labId + ".xml"));
                                          translatedData = mr.ParseMadsFile(xtr, tht, labId);

                                          ldf.insertEarssData(translatedData);
                                          break;

                                    case "xls":
                                          ldf = new LD_Facade();
                                          ADBaktReader ar = new ADBaktReader(ldf.getExcelWorkbook(filePath));
                                          tht = (new HashmapCreator().createHashmaps("../TranslationFiles/" +
labId + ".xml"));
                                          translatedData = ar.parseExcel(tht,columnTable,labId);
                                          ldf.insertEarssData(translatedData);
                                          break;
                              }
                        }
                        else{
                              ErrorHandler.AddToErrorLog("TranslationEngine.TranslateData: Error in fileType");
                        }
                  }
            }

            public ArrayList getHeadlines(string filePath){
                  ArrayList headlines = new ArrayList();

                  ADBaktReader ar = new ADBaktReader(ldf.getExcelWorkbook(filePath));
                  headlines = ar.returnHeadlineValues();

                  return headlines;
```

```
            }
      }
}
// End of TranslationEngine


-------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Xml;
using System.Collections;
using EarssDataConversion.DataClasses;
using EarssDataConversion.Facades;

namespace EarssDataConversion.DataInsertion
{
      /// <summary>
      /// Description of LabParser.
      /// </summary>
      public class LabParser
      {
            private const string filePath = "../../ConfigFiles/Laboratories.xml";
            public LabParser(){     }

            public ArrayList parseLabs(){
                  LD_Facade ldf = new LD_Facade();
                  ArrayList lablist = new ArrayList();
                  XmlTextReader xtr = ldf.getXmlTextReader(filePath);

                  if (xtr!= null){

                        while ( xtr.Read()){
                              switch(xtr.NodeType){
```

```
                                    case XmlNodeType.Element:
                                          switch (xtr.Name){
                                                case "Name":
                                                      LabShort lab = new LabShort();
                                                      lab.getSetName = xtr.ReadString();
                                                      xtr.Read();xtr.Read();xtr.Read();
                                                      lab.getSetEarssId = xtr.ReadString();
                                                      lablist.Add(lab);


                                                      break;
                                          }
                                          break;
                                    }
                              }
                        }
                        return lablist;
                  }
            }
}
// End of LabParser


------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Text;
using System.IO;
using EarssDataConversion.DataConnections;

namespace EarssDataConversion.ErrorHandling
{
      public static class ErrorHandler
```

```
	{
		private static StringBuilder sb = null;
		private static string errorString = "Errorlog: " + "\r\n";


		public static void AddToErrorLog(string ErrorMsg){
			if (sb == null)
				sb = new StringBuilder();
			else
				sb.Append(ErrorMsg + "\r\n");
		}

		public static void saveErrorLog(){
			TextFileWriter tfw = new TextFileWriter();
			string filePath = Directory.GetCurrentDirectory() + "/error_log.txt";

			if (sb == null){
				tfw.writeFile(filePath, errorString + "No errors!");
			}
			else{
				tfw.writeFile(filePath, errorString + sb.ToString());
			}
		}
	}
}
// End of ErrorHandler


--------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;
using System.Xml;
```

```
using Excel;
using Npgsql;
using EarssDataConversion.DataConnections;

namespace EarssDataConversion.Facades
{
	public class LD_Facade
	{
		public LD_Facade()
		{
		}

		public Excel.Workbook getExcelWorkbook(string filePath){
			ExcelFileReader exf = new ExcelFileReader();
			return exf.ReadExcelFile(filePath);
		}

		public XmlTextReader getXmlTextReader(string filePath){
			XmlFileReader xfr = new XmlFileReader();
			return xfr.getXmlTextReader(filePath);
		}

		public NpgsqlDataReader getEarssData(string query){
			DB_Connection dbc = new DB_Connection();
			return dbc.getEarssData(query);
		}

		public void writeTextFile(string filePath, string fileContent){
			TextFileWriter tfw = new TextFileWriter();
			tfw.writeFile(filePath, fileContent);
		}

		public void insertEarssData(ArrayList translatedData){
			DataInserter di = new DataInserter();
			di.insertData(translatedData);
		}
	}
}
```

```
// End of LD_Facade


-----------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;
using Npgsql;
using EarssDataConversion.Presentation;
using EarssDataConversion.DataInsertion;
using EarssDataConversion.DataExtraction;
using EarssDataConversion.DataInsertion.Translation;

namespace EarssDataConversion.Facades
{
	/// <summary>
	/// Description of PL_Facade.
	/// </summary>
	public class PL_Facade
	{

		public PL_Facade()
		{}


		public ArrayList getHeadlines(string filePath){
			TranslationEngine te = new TranslationEngine();
			return te.getHeadlines(filePath);
		}

		public void insertData(Hashtable columns, string filePath, string labId){
			TranslationEngine te = new TranslationEngine();
			te.TranslateData(columns, filePath, labId);
```

```
            }

            public ArrayList getLabs(){
                  LabParser lp = new LabParser();
                  ArrayList list = new ArrayList();
                  list = lp.parseLabs();
                  return list;
            }

            public void extractEarssData(string filePath, string year, string quarter, ArrayList pathogens){
                  ExtractionEngine ee = new ExtractionEngine();
                  ee.extractEarssData(filePath, year, quarter, pathogens);
            }
      }
}
// End of PL_Facade


-------------------------------------------------------------------------------
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Collections;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Npgsql;
using Excel;
using EarssDataConversion.DataClasses;
using EarssDataConversion.Facades;

namespace EarssDataConversion.Presentation
{
      /// <summary>
```

```
/// Description of MainForm.
/// </summary>
public partial class MainForm
{
	PL_Facade plf = null;
	public MainForm()
	{

	}

	public MainForm(PL_Facade plf)
	{
		this.plf = plf;
		InitializeComponent();
	}

	void btn_CloseClick(object sender, System.EventArgs e)
	{
		Close();
	}

	void cb_LabsSelectedIndexChanged(object sender, System.EventArgs e)
	{

	}

	void MainFormLoad(object sender, System.EventArgs e)
	{
		this.fillComboKMA();
		this.fillExtractionCombos();

	}

	void btn_saveLabClick(object sender, System.EventArgs e)
	{

	}
```

```
		void Btn_updateLabClick(object sender, System.EventArgs e)
		{

		}

		void Btn_deleteLabClick(object sender, System.EventArgs e)
		{

		}

		void btn_OpenFileClick(object sender, System.EventArgs e)
		{
			openFileDialog1 = new OpenFileDialog();
			openFileDialog1.ShowDialog();
			txt_dataFileName.Text = openFileDialog1.FileName;

		}


		void btn_HeadlinesClick(object sender, System.EventArgs e)
		{
			if (txt_dataFileName.Text != ""&& cb_insertData_labs.Text != ""){
				ArrayList headlines = new ArrayList();
				headlines = plf.getHeadlines(txt_dataFileName.Text);
				ArrayList combos = getCombos();

				foreach (ComboBox c in combos){
					foreach (String s in headlines){
						c.Items.Add(s.ToString());
					}
				}
			}
			else
				MessageBox.Show("Vælg venligst en datafil der skal indlæses" + "\n" +" og hvilken KMA
filen kommer fra!");
		}

		void btn_extractDataClick(object sender, System.EventArgs e)
```

```
{
	string year = cb_extractYear.Text;
	string quarter = cb_extractQuarter.Text;
	ArrayList pathogens = new ArrayList();

	if(chb_eco.Checked == true){
		pathogens.Add("eco");
	}
	if(chb_spn.Checked == true){
		pathogens.Add("spn");
	}
	if(chb_sau.Checked == true){
		pathogens.Add("sau");
	}
	if(chb_efa.Checked == true){
		pathogens.Add("efa");
	}
	if(chb_efm.Checked == true){
		pathogens.Add("efm");
	}
	if(chb_kpn.Checked == true){
		pathogens.Add("kpn");
	}
	if(chb_pae.Checked == true){
		pathogens.Add("pae");
	}

	if(txt_outputFile.Text == ""){
		MessageBox.Show("Husk at vælge et navn til filen");

	}
	else
	if(pathogens.Count<1)
		MessageBox.Show("Vælg mindst en bakterie");

	else
		if(cb_extractQuarter.SelectedIndex == -1)
			MessageBox.Show("Vælg kvartal");
```

```
                else
                      if(cb_extractYear.SelectedIndex == -1)
                            MessageBox.Show("Vælg årstal");
                else
                      //MessageBox.Show(txt_outputFile.Text + " " + cb_extractYear.SelectedItem.ToString() + "
" +(cb_extractQuarter.SelectedIndex+1).ToString());
                      plf.extractEarssData(txt_outputFile.Text, cb_extractYear.SelectedItem.ToString()
,(cb_extractQuarter.SelectedIndex+1).ToString(), pathogens);

          }

          void Cb_insertData_labsSelectedIndexChanged(object sender, System.EventArgs e)
          {

                switch(cb_insertData_labs.SelectedItem.ToString()){

                      case "Rigshospitalet":
                                              enableCombos();
                                              break;

                      case "Hvidovre":
                                              enableCombos();
                                              break;

                      case "Herlev":
                                              enableCombos();
                                              break;

                      case "Hillerød":
                                              enableCombos();
                                              break;

                      case "Roskilde":
                                              enableCombos();
                                              break;

                      case "Slagelse":
                                              enableCombos();
```

```
                        break;
        case "Næstved":
                        disableCombos();
                        break;
        case "Odense":
                        enableCombos();
                        break;
        case "Sønderborg":
                        enableCombos();
                        break;
        case "Esbjerg":
                        enableCombos();
                        break;
        case "Vejle":
                        enableCombos();
                        break;
        case "Herning":
                        enableCombos();
                        break;
        case "Skejby":
                        disableCombos();
                        break;
        case "Viborg":
                        enableCombos();
                        break;
        case "Aalborg":
                        enableCombos();
                        break;
}
```

```
}

void btn_ReadClick(object sender, System.EventArgs e)
{
      if (txt_dataFileName.Text != ""&& cb_insertData_labs.Text != ""){
            Hashtable ht = new Hashtable();

            if (btn_Headlines.Enabled == true){

                  ht.Add("abCode", cb_abCode.SelectedIndex + 1);
                  ht.Add("admisDate", cb_admisDate.SelectedIndex + 1);
                  ht.Add("birthMonth", cb_birthMonth.SelectedIndex + 1);
                  ht.Add("birthYear", cb_birthYear.SelectedIndex + 1);
                  ht.Add("clinDiag", cb_clinDiag.SelectedIndex + 1);
                  ht.Add("collDate", cb_collDate.SelectedIndex + 1);
                  ht.Add("diskLoad", cb_diskLoad.SelectedIndex + 1);
                  ht.Add("esbl", cb_esbl.SelectedIndex + 1);
                  ht.Add("etestGtLt", cb_etestGtLt.SelectedIndex + 1);
                  ht.Add("etestValue", cb_etestValue.SelectedIndex + 1);
                  ht.Add("hospCode", cb_hospCode.SelectedIndex + 1);
                  ht.Add("hospDep", cb_hospDep.SelectedIndex + 1);
                  ht.Add("micGtLt", cb_micGtLt.SelectedIndex + 1);
                  ht.Add("micValue", cb_micValue.SelectedIndex + 1);
                  ht.Add("origin", cb_origin.SelectedIndex + 1);
                  ht.Add("othDiag", cb_othDiag.SelectedIndex + 1);
                  ht.Add("pathogen", cb_pathCode.SelectedIndex + 1);
                  ht.Add("patientId", cb_patientId.SelectedIndex + 1);
                  ht.Add("pbp", cb_pbp.SelectedIndex + 1);
                  ht.Add("pcrMec", cb_pcrMec.SelectedIndex + 1);
                  ht.Add("sampleId", cb_sampleId.SelectedIndex + 1);
                  ht.Add("seroType", cb_serotype.SelectedIndex + 1);
                  ht.Add("sex", cb_sex.SelectedIndex + 1);
                  ht.Add("sir", cb_sir.SelectedIndex + 1);
                  ht.Add("source", cb_source.SelectedIndex + 1);
                  ht.Add("zoneGtLt", cb_zoneGtLt.SelectedIndex + 1);
                  ht.Add("zoneValue", cb_zoneValue.SelectedIndex + 1);
            }
            plf.insertData(ht, txt_dataFileName.Text, cb_insertData_labs.Text);
```

```
        }
}

private void fillComboKMA()
{
        ArrayList labs = new ArrayList();
        labs = plf.getLabs();

        foreach (LabShort lab in labs)
                cb_insertData_labs.Items.Add(lab.getSetName);
}

private void fillExtractionCombos(){
        cb_extractYear.Items.Add("2005");
        cb_extractYear.Items.Add("2006");
        cb_extractQuarter.Items.Add("1. Kvartal");
        cb_extractQuarter.Items.Add("2. Kvartal");
        cb_extractQuarter.Items.Add("3. Kvartal");
        cb_extractQuarter.Items.Add("4. Kvartal");
}

private ArrayList getCombos(){
        ArrayList comboBoxes = new ArrayList();

        comboBoxes.Add(cb_abCode);
        comboBoxes.Add(cb_admisDate);
        comboBoxes.Add(cb_birthMonth);
        comboBoxes.Add(cb_birthYear);
        comboBoxes.Add(cb_clinDiag);
        comboBoxes.Add(cb_collDate);
        comboBoxes.Add(cb_diskLoad);
        comboBoxes.Add(cb_esbl);
        comboBoxes.Add(cb_etestGtLt);
        comboBoxes.Add(cb_etestValue);
        comboBoxes.Add(cb_hospCode);
        comboBoxes.Add(cb_hospDep);
        comboBoxes.Add(cb_micGtLt);
        comboBoxes.Add(cb_micValue);
```

```
                  comboBoxes.Add(cb_origin);
                  comboBoxes.Add(cb_othDiag);
                  comboBoxes.Add(cb_pathCode);
                  comboBoxes.Add(cb_patientId);
                  comboBoxes.Add(cb_pbp);
                  comboBoxes.Add(cb_pcrMec);
                  comboBoxes.Add(cb_sampleId);
                  comboBoxes.Add(cb_serotype);
                  comboBoxes.Add(cb_sex);
                  comboBoxes.Add(cb_sir);
                  comboBoxes.Add(cb_source);
                  comboBoxes.Add(cb_zoneGtLt);
                  comboBoxes.Add(cb_zoneValue);

                  return comboBoxes;
            }

            private void disableCombos(){
                  btn_Headlines.Enabled = false;
                  ArrayList combos = getCombos();
                  foreach (ComboBox c in combos){
                        c.Enabled = false;
                  }
            }

            private void enableCombos(){
                  btn_Headlines.Enabled = true;
                  ArrayList combos = getCombos();
                  foreach (ComboBox c in combos){
                        c.Enabled = true;
                  }
            }
      }
}
// End of MainForm


--------------------------------------------------------------------------------
```

```
/*
 * Created by SharpDevelop.
 * User: SSY
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Windows.Forms;
using EarssDataConversion.ErrorHandling;
using EarssDataConversion.Facades;
using EarssDataConversion.Presentation;

namespace EarssDataConversion
{
	/// <summary>
	/// Description of MainClass.
	/// </summary>
	public class MainClass
	{
		[STAThread]
		public static void Main(string[] args)
		{
			PL_Facade plf = new PL_Facade();
			Application.EnableVisualStyles();
			Application.SetCompatibleTextRenderingDefault(false);
			Application.Run(new MainForm(plf));

			ErrorHandler.saveErrorLog();
		}

		public MainClass()
		{
		}
	}
}
// End of MainClass
```